REAL-TIME OPTIMAL CONTROL OF AUTONOMOUS SWITCHED SYSTEMS

A Thesis
Presented to
The Academic Faculty

by

Xu Chu Ding

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2009

REAL-TIME OPTIMAL CONTROL OF AUTONOMOUS SWITCHED SYSTEMS

ACKNOWLEDGEMENTS

First and foremost, I would like to express my appreciation to my advisors, Dr. Magnus Egerstedt and Dr. Yorai Wardi, for their guidance and support, without which this dissertation would not have materialized. Your unquenchable enthusiasm and tireless hard-work have been the most invaluable encouragement to me. I also wish to thank Dr. David Taylor and Dr. Anthony Yezzi for serving on my reading committee, and Dr. George Riley and Dr. Craig Tovey for being on my defense committee.

The collaboration with Axel Schild on the topic of state-feedback optimal control of switched systems is fruitful and has resulted in an important part of this dissertation. The pilot decision support framework is resulted from the delightful collaboration with Matt Powers. I also enjoy the collaboration with Dr. Amir Rahmani on the topic of convoy protection optimal control. Furthermore, I must thank my fellow students in the GRITS Lab for their friendship and many meaningful conversations: Rahul Chipalkatty, Musad Haque, Jonghoek Kim, Peter Kingston, Patrick Martin, Amir Rahmani, Axel Schild and Philip Twu (sorted by last name).

Last but not least, I would like to thank my parents for their endless love and unwavering support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis provides a real-time algorithmic optimal control framework for autonomous switched systems. Traditional optimal control approaches for autonomous switched systems are open-loop in nature. Therefore, the switching times of the system can not be adjusted or adapted when the system parameters or the operational environments change. This thesis aims to close this loop, and apply adaptations to the optimal switching strategy based on new information that can only be captured on-line. One important contribution of this work is to provide the means to allow feedback (in a general sense) to the control laws (i.e. the switching times) of the switched system so that the control laws can be updated to maintain optimality of the switching-time control inputs. Furthermore, convergence analyses for the proposed algorithms are presented. The effectiveness of the real-time algorithms is demonstrated by an application in optimal formation and coverage control of a networked system. This application is implemented on a realistic simulation framework consisting of a number of Unmanned Aerial Vehicles (UAVs) that interact in a virtual 3D world.

CHAPTER I

INTRODUCTION

1.1 Introduction to Hybrid and Switched Systems

As control systems grow more complex, general system theories become less adequate to study and regulate the behaviors of these systems. In the last few decades, there has been a huge amount of research on a special class of systems, which are called *hybrid systems*. Hybrid systems can be described as systems that are controlled by discrete events in the higher level, while their dynamical behaviors are governed by continuous dynamical laws in the lower level. A hybrid system is hybrid in nature because they obey a set of possible dynamical laws. For such a hybrid system, discrete events in the higher level dictate the continuous dynamical behavior of the system in the lower level.

Hybrid systems appear in many applications and technologies, since there are a huge number of systems that permit a number of modes of operations in their behavior. These systems can be naturally modelled as hybrid systems. Furthermore, it is easier to control many systems by dividing their control laws into distinct modes than traditional control strategies. As a result, hybrid system control problems arise in a variety of applications, including situations where a control module has to switch its attention among a number of subsystems [49, 59, 67], collect data sequentially from a number of sensory sources [22, 45], and achieve formation and coverage control of networked systems [10, 12]. Relevant domains that require hybrid system control theories include the field of behavior-based robotics [5, 9, 38], manufacturing systems [53], power electronics [24, 29, 40], multi-agent network control [56] and air traffic control [66]. By no means is the above list complete or exhaustive, since hybrid systems by now are widely used in many fields and applications. See [41] and reference therein for a recent survey on hybrid systems.

General nonlinear control theories without modification usually do not work well with hybrid systems since they do not capture the salient structures of distinct modes of operation. This is due to the fact of hybrid systems being controlled by switching strategies in addition to continuous control signals. A switching strategy for a hybrid system provides the appropriate continuous dynamical law for the system, as well as conditions to change from one dynamical law to another (discrete events). Effective control of a hybrid system requires an adequate switching strategy as well as a suitable control input signal.

We preface the discussion of hybrid and switched systems with a brief historical perspective. The interest in hybrid automata and hybrid systems boomed in the 1980s due to increasingly accessible methods to analyze these systems. In early 1990s, embedded systems that utilize event driven dynamics in the upper level, and digitized controllers that control continuous systems in the lower level, became more common. ATM machines and common house appliances are instances of such systems. All of these systems are examples of hybrid systems, and the effective control of these systems became very important. Many studies then surfaced to model and analyze these hybrid systems.

In [21] several models of hybrid systems were proposed and a framework was provided to study these hybrid systems. Reference [20] provided a unified framework to model and control hybrid systems, by proposing two different methodologies (or strategies) for controlling hybrid systems: autonomous switching and controlled switching. Stability analysis of hybrid systems was investigated through different approaches, for example [19, 46, 70]. Recently, many research groups started to focus on optimal control of hybrid systems, and these studies led to real-time optimal control of switched systems, which is the content of this thesis. We withhold a more detailed discussion on optimal control of hybrid systems until Chapter 2.

1.1.1 Behavior and Control of Hybrid Systems

To understand the behavior and control of hybrid systems, we first introduce the notion of *modes*. Each mode of a hybrid system corresponds to a dynamical law. A *mode sequence* is defined as a sequence of modes. *Mode switching* (or *mode transition*) refers to switching the mode of operation of the system from the current mode to the next mode according to the mode sequence. Using the framework proposed in [20], a hybrid system can be modelled as a system that allows itself to be controlled by either autonomous or controlled switching strategies, or both.

Autonomous switching is a switching strategy that is characterized by switching manifolds (or more generally, regions of state space), which trigger mode transitions. The switching manifolds are previously given and fixed, and a mode transition from one vector field to another is defined for each manifold. In the literature, this class of hybrid systems is commonly referred to as *autonomous switching hybrid systems* (ASHS). The switching times associated with an ASHS are called autonomous switching times.

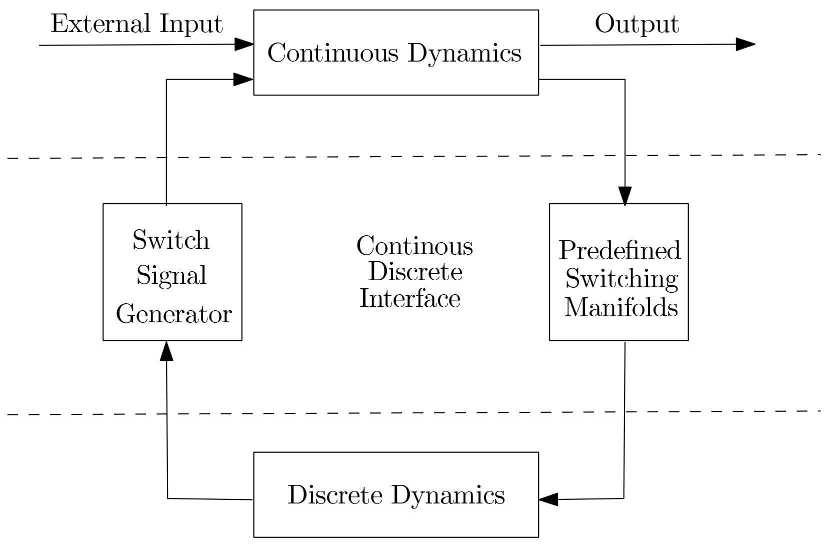


Figure 1: Block diagram model of an autonomous switching hybrid system.

The model of an ASHS, demonstrated using block diagrams, is shown in Figure 1. As seen from Figure 1, it is useful to separate the interface between the discrete and the continuous component of a hybrid system. In the lower level, the continuous dynamics is controlled by an external input and a switch signal commanded by a switch signal generator. In the higher level, the discrete dynamics is controlled by the predefined switching manifold (or switching regions). If the state of the system is within the switching manifold, the discrete dynamics (controlled according to a suitable mode sequence) triggers a mode transition by generating a switch signal. The switch signal then enforces the system to change its dynamical law to the desired one encoded in the mode sequence. It should be noted that, since the switching manifolds (or regions) are fixed a priori, an ASHS is only controlled by the external input.

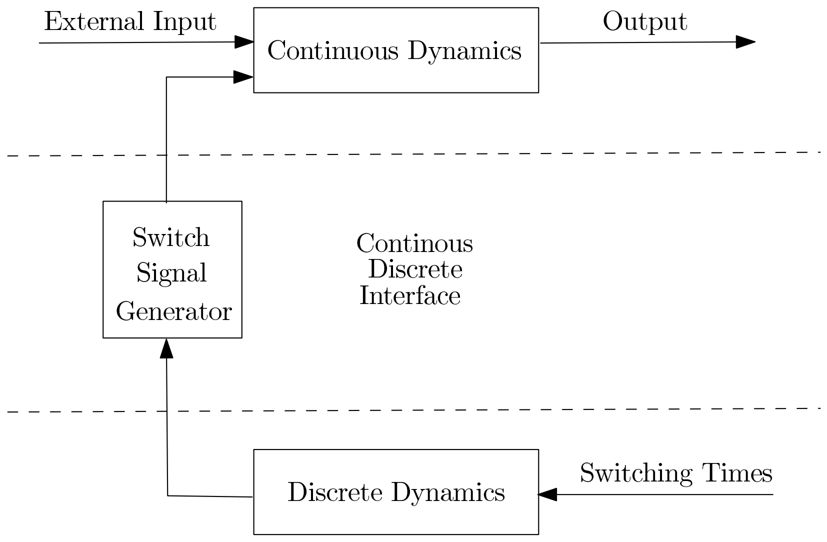


Figure 2: Block diagram model of a controlled switching hybrid system.

Controlled switching is a switching strategy that is characterized by a sequence of switching time instants. Hence, it is assumed that there is an accurate system clock embedded within the hybrid system that checks its time with the desired switching times. This class of hybrid system is referred to as *controlled switching hybrid systems* (CSHS). The switching times associated with a CSHS are called controlled switching times. A model of a CSHS, demonstrated using block diagrams, is shown in Figure 2. Similar to an ADHS, switch transitions are triggered by switch signals generated by the discrete dynamics, which enforce the system to switch its dynamical law according to a fixed mode sequence.

Combining both autonomous and controlled switching strategies, a general hybrid system can be modelled using block diagrams as shown in Figure 3.

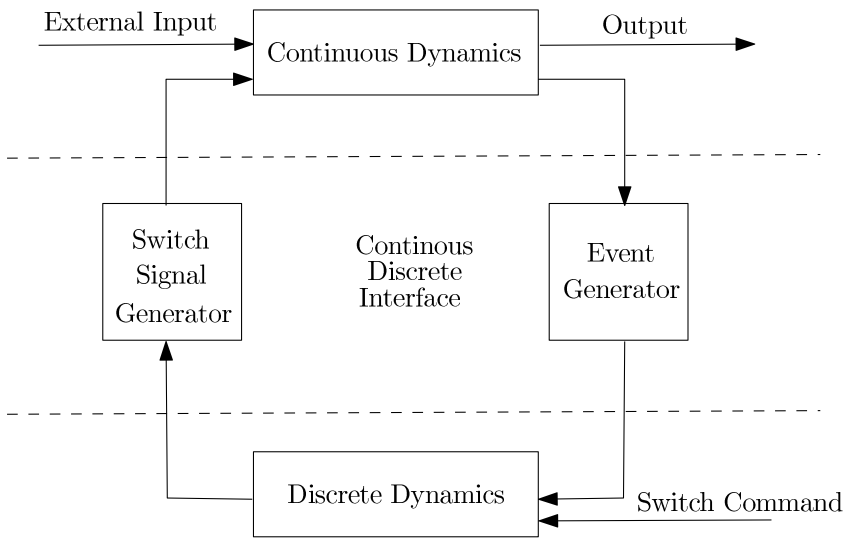


Figure 3: Block diagram model of a general hybrid system.

Furthermore, a hybrid system may undergo a discrete state jump at each mode transition. State jumps are modelled by a (possibly nonlinear) function defined on the state of the system for each mode. When the hybrid system triggers a mode switching to the next mode, the state at that time instant is reset using the state jump function corresponding to the new mode.

Finally, if a hybrid system does not allow an external control input, then it is called an autonomous hybrid system. Note that, if an ASHS is autonomous, then it is fully autonomous in the sense that there is no external control influence on the system. However, if a CSHS is autonomous, then the system is controlled by the sequence of switching times, and this sequence composes the sole control influence.

1.1.2 Switched Systems

Switched system is a term used in many hybrid system literature as a short name for controlled switching hybrid systems. Autonomous switched systems are autonomous CSHS, hence they do not allow an external control input. We also use this definition in this thesis, and this thesis focuses on autonomous switched systems.

It should be noted that incorporating ASHS into the general control theory of switched systems typically requires a straight-forward extension, since switching manifolds are fixed a priori and they are not part of the control variables of the system. For optimal control problems, this extension causes the costate Lagrange Multiplier trajectories to contain discrete jumps at switching times instead of being continuous (see [63]).

1.2 Mathematical Model of Autonomous Switched Systems

We now establish a mathematical description of autonomous switched systems and introduce a set of basic notations and terminologies which will be used throughout this thesis.

We denote $x(t) \in \mathbb{R}^n$ as the state of the system at time t. Using the models established in the previous section, we can describe the evolution of an autonomous switched system using the differential inclusion form

$$\dot{x} \ \in \ \{f_q(x),\ q \in \mathcal{A}\}, t \in [0,T], \tag{1}$$

with a time horizon T and initial condition x_0. $\mathcal{A}$ is a finite index set enumerating the possible modes of the system, and $\{f_q : \mathbb{R}^n \to \mathbb{R}^n\}_{\{q \in \mathcal{A}\}}$ is a collection of (generally nonlinear) sufficiently smooth functions. In this thesis, we usually assume that the modal functions are continuously differentiable, and we will note their smoothness requirements in the proposed results. The functions f_q, $q \in \mathcal{A}$ are henceforth referred to as *modal functions*.

The mode sequence of the system is denoted as $\bar{q} = \{q_1, q_2, ..., q_{N+1}\}$, where $N+1$ is the total number of modes. Note that the mode sequence $\bar{q}$ and hence N are not required to be fixed. The switching time instant between mode i and mode $i+1$ is denoted by τ_i. Given the mode sequence of the system, we call the vector $\bar{\tau} = [\tau_1, \tau_2, ..., \tau_N]^T$ the *switching time vector*, and as such $\mathcal{M} = \{\bar{q}, \bar{\tau}\}$ forms a *switching schedule* for the system. Note that the switching schedule $\mathcal{M}$ composes the entire control influence on an autonomous switched system.

For all switching time vectors in $\mathbb{R}^N$, a feasible switching time vector must follow the piecewise linear inequality constraint

$$0 = \tau_0 \leq \tau_1 \leq ... \leq \tau_N \leq \tau_{N+1} = T. \tag{2}$$

We define feasible set Λ as the set of all switching time vectors that satisfy (2), hence, $\Lambda := \{\bar{\tau} = [\tau_1, \tau_2, ..., \tau_N]^T : 0 = \tau_0 \leq \tau_1 \leq ... \leq \tau_N \leq \tau_{N+1} = T\}$. A switching time vector $\bar{\tau}$ is a feasible control input for an autonomous switched system if and only if $\bar{\tau} \in \Lambda$.

To incorporate possible state jumps at mode transitions, we may define $\Psi_i : \mathbb{R}^n \to \mathbb{R}^n, i = 1, 2, ..., N+1$ as a collection of state jump functions, corresponding to a mode sequence $\bar{q}$. For the i-th mode switching of the system, the state of the system after mode transition is reset by the state jump function as

$$x(\tau_i^+) = \Psi_{i+1}\left(x(\tau_i^-)\right), i = 1, 2, ..., N, \tag{3}$$

where $x(\tau_i^+)$ denotes the state of the system immediately after the jump and $x(\tau_i^-)$ immediately before the jump. To simplify the control problem and notations, we do not consider state jumps in this thesis. Therefore, the state of the system is always continuous across switching times.

For simplicity of notation, we define $\tau_0 = 0$ and $\tau_{N+1} = T$ and denote $f_i(x) = f_{q_i}(x)$. Then, given a switching schedule $\mathcal{M}$, we can describe the evolution of an autonomous switched system in the following form.

$$\dot{x} = f_i(x), t \in [\tau_{i-1}, \tau_i), i = 1, 2, ..., N + 1. \tag{4}$$

Equation (4) describes the mathematical model of autonomous switched systems we use throughout this thesis. Figure 4 illustrates an example trajectory of a scalar valued autonomous switched system.

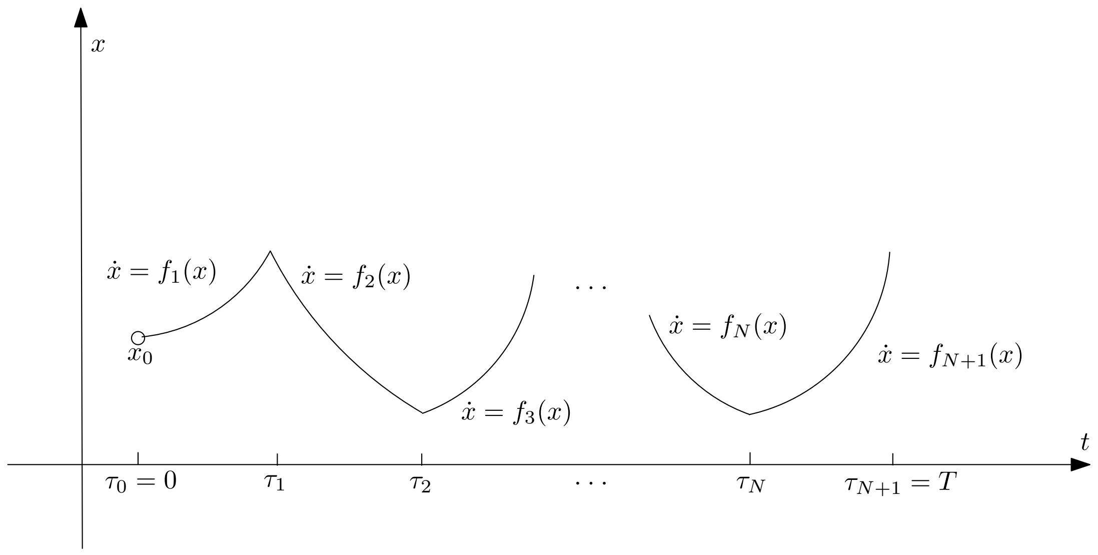


Figure 4: Example of a scalar switched system ($x(t) \in \mathbb{R}$). The system switches from the mode i to mode $i + 1$ at time τ_i.

1.3 Optimal Control of Autonomous Switched Systems

Optimal control is deep-rooted in general control theory, and it is particularly useful for complex and nonlinear systems, such as hybrid systems. Thus, much research attention has been focused on establishing an optimal control framework for hybrid systems. Optimal control allows one to quantify the performance of a system using a cost defined on the state trajectory of the system, and the control problem becomes finding the control input that optimizes this performance (or equivalently, minimizes the cost). To this end, let $L_i : \mathbb{R}^n \to \mathbb{R}$ and $\phi_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, 2, ..., N + 1$, to be a collection of (generally nonlinear) sufficiently smooth functions, and then we define the cost functional J as the following

$$J = \sum_{i=1}^{N+1} \left(\int_{\tau_{i-1}}^{\tau_i} L_i(x(t))dt + \phi_i\left(x(\tau_i)\right) \right). \tag{5}$$

The function L_i is called a trajectory cost function and ϕ_i is called a switching cost function. They are usually assumed to be continuously differentiable, and again we will note their smoothness requirements in the proposed results.

Optimal control of autonomous switched systems involves minimizing the cost functional J over switching schedules $\mathcal{M} = \{\bar{q}, \bar{\tau}\}$. This problem is called *optimal timing control problem.* The switching schedule $\mathcal{M}$ consists of two variables: the sequence variable $\bar{q}$ and the timing variable $\bar{\tau}$. Hence, this problem can be further divided into two sub-problems. The problem of finding optimal placement of switching times given a fixed switching sequence is referred in the literature as the *time optimization problem.* This is the easier problem of the two, since it can be viewed as a nonlinear finite dimensional optimization problem. The problem of finding an optimal switching sequence of a switched system is a more challenging problem, due to the variable dimension of the control variable. This optimization problem involves optimizing the cost functional J over the space of variable-length sequences, which is generally a hard problem. This is called *optimal mode scheduling problem.*

Note that the switching cost function ϕ_i is optional for an optimal control problem, and they are sometimes omitted in the problems considered in this thesis. Extension of the results to include switching costs is generally possible, and an example can be seen in [29].

It should be stressed that, the optimization problem at hand is generally nonconvex, since $J(\bar{\tau})$, given $\bar{q}$, is not a convex function in general. Hence, the goal of the optimization problem is to find a *local minimum* that satisfies certain necessary optimality conditions. Thus, all algorithms presented in this thesis are limited to finding local optimal solutions.

1.4 Motivations and Contributions

An abundant amount of work has been done for the optimal control problem of autonomous switched systems. A number of approaches that pertain to this problem can be found in [7, 8, 14, 15, 18, 39, 43, 44, 48, 49, 58, 62, 63, 64, 70, 71, 72] and references therein (these approaches form the necessary foundation for this thesis, and they will be discussed more in detail in Chapter 2). However, it should be made clear that an optimal control problem formulated in the last section aims to find a *feed-forward* control, which is the switching times that minimize the cost functional J. This is not surprising, because optimal control approaches for nonlinear systems generally generate *open-loop* control input. In this sense, all of these approaches address the problem in the *off-line* setting, since the optimal solution is computed off-line, and then it is applied to the plant without modification during run-time of the system. Furthermore, the optimal solution computed with these approaches requires and depends on the initial condition of the system.

There are a number of issues with this open-loop approach of controlling switched systems. Some applications simply can not stay idle and wait for off-line computations. The performance of the system in run-time may be degraded by noise and disturbances, and initial condition of the system may be unknown. These problems are currently rarely addressed in the literature. This thesis aims to address these problems that emerge during real-time operation of the plant, and the goal of this thesis is to provide a novel and original framework for optimal control of autonomous switched systems in real-time. One important contribution of this thesis is to provide the means to allow *feedback* (in a general sense) to the control laws of the switched system so that the control law (switching times) can be updated to maintain optimality.

An example of this feedback-based control framework is in the form of using updated state estimates obtained through observer feedback of the switched system.

Assume that there is a state observer that provides increasingly accurate state information and the optimal control laws of the system must be updated so that they remain optimal with respect to improving state estimates. Using this method, the initial state of the switched system is not required to be known before the system starts running, which is important for most realistic applications. Another example is when a switched system is used to track a moving target that can only be measured in real time. In this case, the cost functional needs to be constantly updated.

In addition, this thesis addresses the problem when the switched system operates under a noisy environment. Hence, this thesis provides an optimal controller which is robust under disturbances. This controller is capable of adjusting according to state measurements and compensating the system for low amplitude and high frequency deviations from the optimal trajectory. This state-feedback controller utilizes optimal switching planes, and compensation of small deviation from an optimal trajectory is achieved by intersecting this plane. This intersection require little computation to check, and thus this approach is highly suitable for disturbances. As an important contribution of this thesis, we show that the optimal switching plane exists and this approach is applicable only if the control problem considered allows the final time T to be part of the control variables.

Finally, optimal control of switched system requires frequent computation of the state trajectory as a function of the control variables. However, when the switched system is required to operate for a long time, it may be no longer feasible to compute the optimal solution due to the length of the simulated state trajectory. In this case, this thesis provides a receding horizon technique to provide sub-optimal control laws for the switched system by using a much smaller time horizon.

An important remark should be made on on-line update of the optimal control input for autonomous switched systems. Due to Bellman's Principle of Optimality, the overall cost will not be affected by using non-optimal control for a period of time, if the optimal switching time is obtained fast enough. This is because the control inputs are in terms of time and they are not applied (by triggering mode transitions) until the operation time of the system reaches the switching time stored in $\bar{\tau}$. This is the reason why on-line optimal control of autonomous switched system is attractive and real-time algorithms proposed in this thesis exploit the special structures of autonomous switched systems. This property is generally not true for non-autonomous switched systems or other nonlinear real-time optimal control problems.

The objective of this thesis is to provide a set of real-time algorithms that generate on-line optimal controllers for autonomous switched systems. The algorithms are required to be feasible to be implemented in real-time. Certainly, this feasibility depends on the computation hardware on-board the system. Hence it is important to first define what constitutes a "real-time" algorithm in this thesis, and this definition needs to be independent of the computation hardware. We consider an algorithm to be real-time if it is executed during the operation of the system, and at each time iteration of the algorithm, the number of forward simulations of the state trajectory must be low enough. We regard this number to be low enough if it is a polynomial function of N (number of switches).

Let us give an example of an algorithm which is not considered real-time in this thesis. One can design a trivial algorithm that recomputes the optimal solution using the current state measurement at each time iteration of the system by formulating a new optimal control problem and solving it using any optimization approaches (say, gradient descent method). This algorithm solves the problems posed by this thesis, but it requires a non-polynomial function of N number of forward simulations to achieve convergence at each time iteration. This is because the optimization problem is highly nonlinear and convergence may take very long amount of time. Worst of all, the performance of this algorithm can not be evaluated because there is no way to know how many forward simulations are required for each iteration.

The algorithms proposed in this thesis execute one step of either gradient descent (first order) or Newton (second order) method at each time iteration. The computation of the gradient requires one forward simulation of the state trajectory, and the computation of the Hessian matrix (using finite difference approximation) requires $\frac{N^2}{2}$ number of simulations. Hence, they can be considered real-time algorithms, and throughout this thesis, we assume that the computation hardware is fast enough to finish these computations in each iteration.

It should be noted that, due to the real-time requirement of the proposed algorithms, this thesis mainly focuses on the time optimization problem (the problem of minimizing J over $\bar{\tau} \in \Lambda$, given a mode sequence $\bar{q}$). Optimal mode scheduling problems are generally much harder to solve, and the algorithms that address these problems are usually highly computationally expensive even in the off-line case. In this thesis, we usually assume that a mode sequence is fixed and given a priori, which is the case for many applications. We propose an algorithm that improves the current mode sequence, but we never regard the final mode sequence as being optimal.

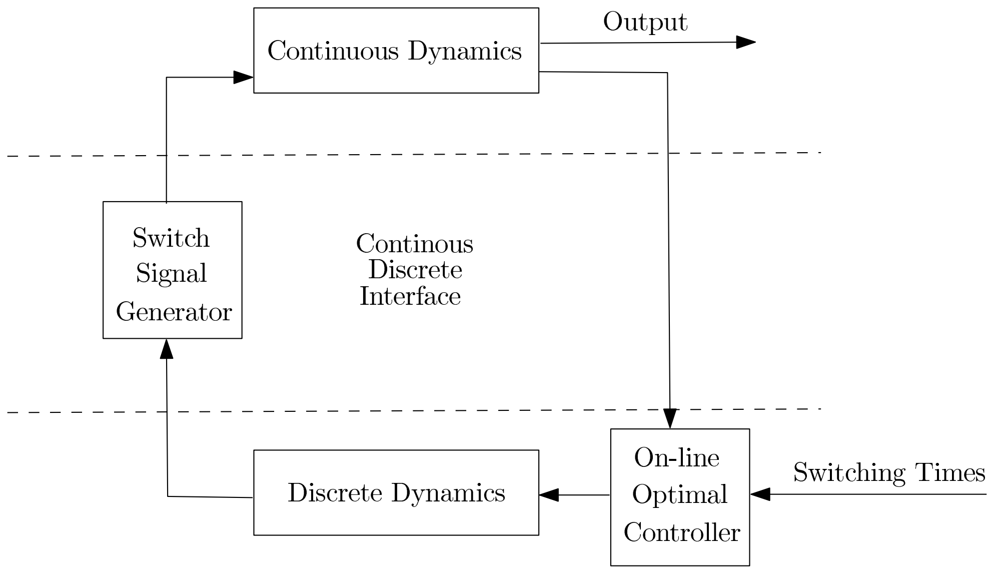


Figure 5: Real-time control of an autonomous switched system. The switching times of the system are modified by the on-line optimal controller based on state feedback.

In summary, this thesis provides a *closed-loop* optimal control framework for autonomous switched systems, i.e. using state feedback to control the switched system in real time. This idea is illustrated in Figure 5 using the block diagram representation of switched systems. This thesis also supplies convergence analysis for the proposed algorithms.

1.5 Organization of the Thesis

The organization of this thesis is as follows: This chapter provides a background introduction to optimal control of autonomous switched systems. In Chapter 2, some previous work on open-loop optimal control of switched systems is presented. These work establishes the groundwork, on which this research is based on. Chapter 3 introduces observer-based optimal timing control of autonomous switched systems [35, 68]. Chapter 4 focuses on a special class of optimal control problems, where an explicit state-feedback optimal switching controller can be obtained [32, 61]. Chapter 5 studies the problem of optimal timing control of autonomous switched systems when the cost function is not known before run-time [33, 34]. Chapter 6 studies optimal timing control of autonomous switched systems using a receding horizon approach [30]. Chapter 7 provides an application for the real-time control approaches examined in this thesis. The application produces a pilot decision framework where a network team of Unmanned Aerial Vehicles (UAVs) can be controlled and navigated by a single pilot. Within this application, the real-time algorithms studied in this chapter are used to provide on-line pilot decision support to aid the human pilot [30, 31].

The content of this thesis has resulted in a number of publications authored or co-authored by the author of this thesis. The references given here are the publications associated with each chapter.

CHAPTER II

OPEN-LOOP OPTIMAL TIMING CONTROL OF AUTONOMOUS SWITCHED SYSTEMS

In this chapter, we briefly overview previous work on optimal timing control of autonomous switched systems. Various approaches to solve this problem by different research groups are examined. Open-loop optimal control lays the foundation for the real-time closed-loop approaches presented in this thesis.

There are a number of approaches pertaining only to switched systems with linear sub-systems [14, 15, 25, 43, 49]. These results are difficult to be applied to this work, since we are interested in optimal control theory for general switched systems. In this chapter, we focus on results and algorithms [8, 39, 62, 63, 64, 71, 72] which are applicable to switched systems with linear or nonlinear sub-systems.

2.1 Switching-time Optimal Control

First, we formulate a timing optimization problem for autonomous switched systems, using the mathematical model introduced in the previous chapter.

Given an autonomous switched system

$$\dot{x} = f_i(x), t \in [\tau_{i-1}, \tau_i), i = 1, 2, ..., N+1, \tag{6}$$

where $x(t) \in \mathbb{R}^n$ is the state variable. Initial condition $x(0) = x_0$ and final time T are given. Continuously differentiable functions $f_i : \mathbb{R}^n \to \mathbb{R}^n$ are modal functions correspond to a fixed mode sequence $\bar{q} = \{1, 2, ..., N+1\}$. The notations $\tau_0 = 0$ and $\tau_{N+1} = T$ are used. The control input of this system is the switching time vector $\bar{\tau} = [\tau_1, \tau_2, ..., \tau_N]^T$, where it must satisfy the constraint $\bar{\tau} \in \Lambda$ and

$$\Lambda := \{\bar{\tau} = [\tau_1, \tau_2, ..., \tau_N]^T : 0 = \tau_0 \leq \tau_1 \leq ... \leq \tau_N \leq \tau_{N+1} = T\}. \tag{7}$$

We can then formulate a timing optimization problem

$$\min_{\bar{\tau}\in\Lambda} J = \int_0^T L(x(t))dt, \tag{8}$$

with a given continuously differentiable trajectory cost function $L : \mathbb{R}^n \to \mathbb{R}$. This optimization problem is nonlinear, finite dimensional and non-convex. Therefore, the solution that one hope to obtain for this problem is only locally optimal.

A major tool in the field of optimal control is the Pontryagin Maximum Principle. Sussmann [64] first derived a variation of the maximum principle for hybrid systems. The hybrid maximum principle leads to necessary optimality conditions, which are essential for solving optimal control problems.

Before we describe the necessary optimality conditions for this problem, we first need to introduce a set of auxiliary mathematical objects. Given a state trajectory $\{x(\cdot)\}_{t=0}^T, x(t) \in \mathbb{R}^n$, we define a trajectory $\{p(\cdot)\}_{t=0}^T, p(t) \in \mathbb{R}^n$ by the differential equation

$$\dot{p}(t) \;=\; -\frac{\partial f_i}{\partial x}^T(x(t))p(t) - \frac{\partial L}{\partial x}^T(x(t)), t \in [\tau_{i-1}, \tau_i], i = N+1, N, \ldots, 1, \tag{9}$$

with the boundary condition $p(T) = 0$. This trajectory is commonly called the *costate* in the optimal control literature. The costate trajectory is computed in a backwards fashion, starts at time $t = T$ and ends at time $t = 0$, given that the entire state trajectory $x(t)$ is available. The costate trajectory is continuous across switching times. The state and costate pair are usually obtained at each iteration of an optimization algorithm. Given an initial condition x_0 and a switching time vector $\bar{\tau}$, the state is first simulated forward, then the costate can be computed backward. Figure 6 shows an examples of the costate trajectory, and visually displays how the state and costate trajectories are computed.

Finally, we define the Hamiltonian (required for the maximum principle) by

$$H(x(t), p(t)) = p^T(t)f_i(x(t)) + L(x(t)), t \in [\tau_{i-1}, \tau_i], i = 1, 2, \ldots N+1. \tag{10}$$

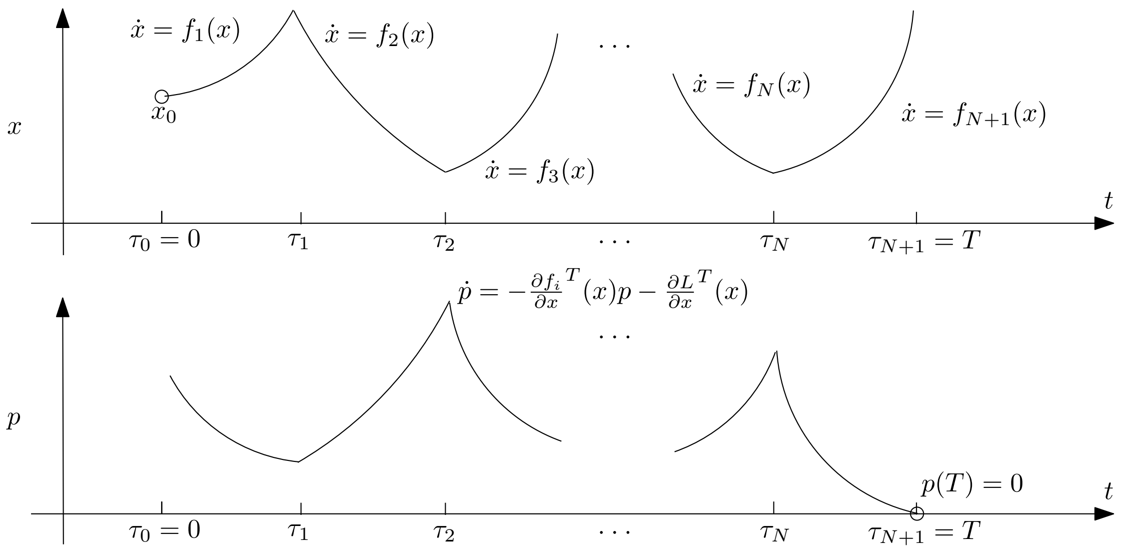


Figure 6: Example of a state and costate pair for a scalar-valued switched system ($x(t) \in \mathbb{R}$). The state trajectory $x(t)$ is computed forward with an initial condition $x(0) = x_0$. Then the costate trajectory $p(t)$ is computed backwards with an initial condition $p(T) = 0$.

The hybrid maximum principle proposed by [64] states that the optimal switching schedule minimizes the Hamiltonian.

2.1.1 First-order Necessary Optimality Conditions and Gradient Formula

Using the hybrid maximum principle as a framework, Shaikh and Caines ([62, 63]) proposed necessary optimality conditions for optimal timing control of switched systems. The necessary optimality conditions state that the solution point for an optimal switching schedule satisfies a set of properties. First, the Hamiltonian is minimized over the space of switching schedules. Second, for a given switching sequence $\bar{q}$, the Hamiltonian is continuous at switching times. In the case of timing optimization problem where a switching sequence $\bar{q}$ is fixed, only the second optimality condition is relevant. Nevertheless, the first optimality condition is useful if one wishes to address the optimal mode scheduling problem.

Using these necessary optimality conditions, multiple algorithms were established by several groups of researchers to solve the timing optimization problem (8). Shaikh and Caines [63] proposed an algorithm that iterates the switching times so that they converge to a solution point where the Hamiltonian is continuous across all switching times, in which case the the optimality condition is fulfilled. This algorithm is similar to the gradient descent algorithm developed by Axelsson, Egerstedt and Wardi in [39]. Namely, both algorithms update the switching times so that the gradient $\frac{\partial J}{\partial \bar{\tau}}$ converges to zero. A number of similar algorithms were proposed by Xu and Antsaklis (see [71, 72]), that use the second derivative of the cost as a function of switching times for the descent direction instead of just the gradient (hence, this approach is using the second order Newton-Raphson method).

In [39], an equivalent necessary optimality condition was derived in the form of

$$\frac{\partial J}{\partial \bar{\tau}} = 0, \text{ and } \frac{\partial^2 J}{\partial \bar{\tau}^2} \succ 0, \tag{11}$$

where the vector

$$\frac{\partial J}{\partial \bar{\tau}} = \left[\frac{\partial J}{\partial \tau_1}, \frac{\partial J}{\partial \tau_2}, ..., \frac{\partial J}{\partial \tau_N}\right]^T \tag{12}$$

is called the gradient and the matrix $\frac{\partial^2 J}{\partial \bar{\tau}^2}$ is called the Hessian.

The equation to compute the gradient for the timing-optimization problem was derived as

$$\frac{\partial J}{\partial \tau_i} = p^T(\tau_i)\bigg(f_i(x(\tau_i)) - f_{i+1}(x(\tau_i))\bigg), i = 1, ..., N. \tag{13}$$

These equations provided a simple method to check for Kuhn-Tucker points of the optimal control problem (they are Kuhn-Tucker points due to the constraint $\bar{\tau} \in \Lambda$). Therefore a gradient descent algorithm can be constructed.

2.1.2 Gradient Descent Algorithm

Before we present a gradient descent algorithm to solve the time optimization problem (8), we should first address the constraint $\bar{\tau} \in \Lambda$. Instead of using the usual approach with Lagrange Multipliers, one should observe that this constraint is linear, and the gradient projection method can be applied. Using this method, we can ignore the constraint, unless the constraint is active and the iteration point is on the boundary of the constraint, in which case the projection of the gradient onto the feasible set Λ should be used as the descent direction. This projection is easy and fast to obtain due to the linearity and simplicity of the constraint, and one can refer to [8], page 26 for an algorithm that projects the gradient vector with any number of N.

We are now ready to present a gradient descent algorithm.

Algorithm 2.1 *Step 0: Choose an initial point $\bar{\tau} \in \Lambda$ and ϵ.*

Step 1: Compute the state trajectory $x(t)$ forward with $\bar{\tau}$ and initial condition x_0, using the dynamical equation (6).

Step 2: Compute the costate trajectory $p(t)$ backward with the terminal condition $p(T) = 0$ and state trajectory $x(t)$, using the differential equation (9).

Step 3: Compute the gradient vector $\frac{\partial J}{\partial \bar{\tau}}$ by equation (13).

Step 4: If $\bar{\tau}$ lies in the interior of Λ, set $\bar{h}(\bar{\tau}) = -\frac{\partial J}{\partial \bar{\tau}}$. Else, set $h(\bar{\tau})$ as the projection of the vector $-\frac{\partial J}{\partial \bar{\tau}}$ onto the feasible set Λ.

Step 5: Compute the Armijo [6] step-size γ.

Step 6: Set $\bar{\tau} = \bar{\tau} + \gamma \bar{h}(\bar{\tau})$.

Step 7: If $||\bar{h}(\bar{\tau})|| < \epsilon$, then quit, otherwise, go back to Step 1.

We now give an example of solving a timing optimization problem by the gradient descent algorithm for an autonomous switched system. Consider a simple current regulator switch circuit as shown in Figure 7. This system can be modelled as a switched system switching between two modes. The state of the system is the current through the inductor.

The goal of this optimal control problem is to optimize the switching times of the circuit so that the current of the circuit tracks a reference current I_{ref}, given a fixed number of switchings. In this example, we fix the number of switchings to 6, and we run Algorithm 2.1 with a random initial value of $\bar{\tau}$. The output optimal solution after the convergence of the gradient descent algorithm is shown in Figure 8.

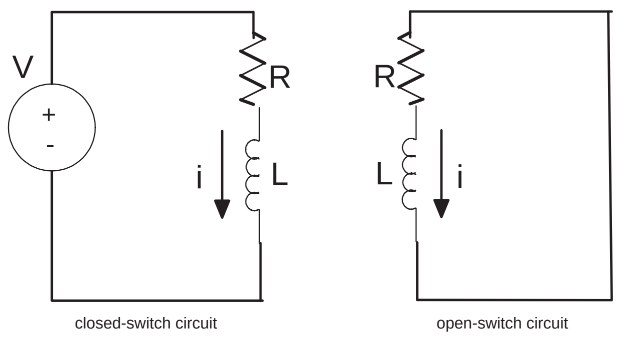


Figure 7: Example of optimal control of a switched system. The system switches from the closed-switch circuit, where the current in the inductor is charged by a power source, to the open-switch circuit, where the current in the inductor is discharged to provide power for a resistive load.

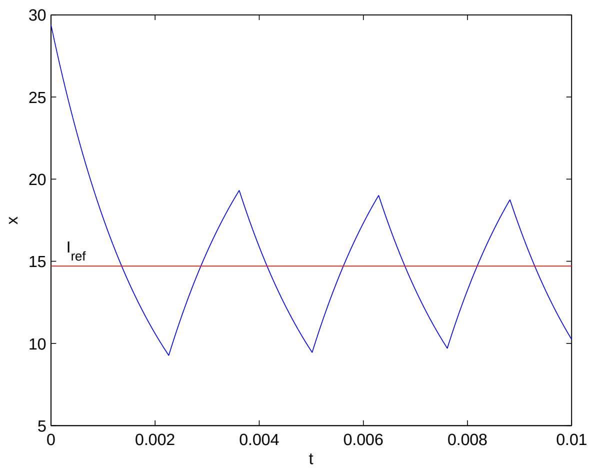


Figure 8: Example of the output of the gradient decent algorithm. x is the trajectory of the current through the inductor. I_{ref} is the current level that the current regulator circuit aims to track. In this example, the current is initially saturated at around 29A, and the I_{ref} is fixed at around 14.5A. The number of switchings is set to 6.

2.2 Mode Scheduling

Now we turn our attention briefly to the problem of mode scheduling, namely the problem of optimizing J as defined in (8) over the space of switching schedules $\mathcal{M} = \{\bar{q}, \bar{\tau}\}$. For this problem, the switching sequence $\bar{q}$ is also part of the control variables, and the number of switchings N is no longer fixed. This problem is much harder, due to the much larger control space consisting of the space of variable-length switching sequences.

As one approach to solve this problem, Shaikh and Caines proposed the concept of an optimality zone (for a detail description, see [62]). The optimality zone is an establishment of a grid that is the Cartesian product of state space coupled with time, times itself $((\mathbb{R}^{n+1})^2)$. First,the optimal state trajectory using the optimal modal function $f_q, q \in \mathcal{A}$ is computed between every point in this grid. Then the algorithm updates the switching times so that the Hamiltonian is continuous at switching transitions, and the optimal state trajectory is obtained from the corresponding points (state coupled with time) on the optimality zone. For this algorithm, both the storage and computational complexity of this method is large, especially for the pre-computation of the optimality zone. Therefore it may not be feasible for a wide range of applications, particularly in a real-time setting.

For the purpose of this thesis, we do not aim to solve the optimal mode scheduling problem in real-time due to high computational complexity of the problem at hand. Instead, we will use the mode insertion idea proposed by Egerstedt, Wardi and Axelsson ([8, 39]) to improve the current switching schedule. This approach is based on needle point variations, and we describe its principle in the next sub-section. This approach will be extended to a real-time setting as explained in the next chapter.

2.2.1 Mode Insertion

Assume that an optimal switching time vector $\bar{\tau}^*$ for the current switching sequence $\bar{q}$ is obtained using Algorithm 2.1. Then one can attempt to add a mode into the current mode sequence and observe the effect of this action on the cost functional J. This process is named *mode insertion*, in which a mode $q \in \mathcal{A}$ is inserted into a different mode (for example mode i) at some time point τ. After each insertion operation, two switching transitions at τ are added to the switching time vector $\bar{\tau}$, and the the mode q is added to the switching sequence $\bar{q}$. This insertion process is illustrated in Figure 9.

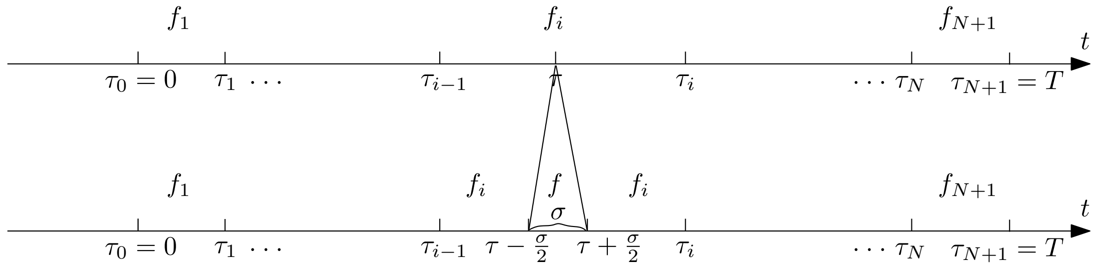


Figure 9: The insertion operation. The mode q is inserted to the i-th mode at time τ for a length of σ. When $\sigma = 0$, $\bar{\tau} = [\tau_1, ..., \tau_i, \tau_{i+1}, ..., \tau_N]$ becomes $[\tau_1, ..., \tau_i, \tau, \tau, \tau_{i+1}, ..., \tau_N]$, and $\bar{q} = \{1, ..., i, i+1, ..., N+1\}$ changes into $\{1, ..., i, q, i+1, ..., N+1\}$.

For the cost functional (8), the inserted switching times incur zero cost right after a mode insertion since the state trajectory $x(t)$ is not changed. For this cost functional, reference [8, 39] proposed a method to evaluate the one-sided derivative of the cost J as a function of the insertion length σ for a non-negative σ. This derivative depends on the time of insertion τ, as well as the modal function inserted (denoted by $f_q, q \in \mathcal{A}$). The evaluation of this derivative for $\sigma = 0$ is called the *insertion gradient*, and it can be computed as

$$\frac{\partial J}{\partial \sigma^+}(0, \tau) = p^T(\tau)(f_i(x(\tau)) - f_q(x(\tau)), \tag{14}$$

where mode q was inserted in mode i. The insertion gradient evaluates the benefit (or lack thereof) of a particular insertion operation. If this insertion gradient is negative, then in a neighborhood of τ, the cost goes down as the insertion length increases (the two inserted points move away from each other). If the gradient descent algorithm is executed for the modified switching schedule, then the cost can be further reduced.

The evaluation of the insertion gradient allows one to find the best time and mode to modify the current switching sequence. Therefore, one can repeatedly solve the timing optimization problem after inserting modes to obtain a better and better switching schedule $\mathcal{M}$. In [8], it was proved that $\mathcal{M}$ is locally optimal if the minimum of insertion gradient is 0 over the time interval $t \in [0, T]$ and the index set $\mathcal{A}$, To make the algorithm practical, the mode insertion stops when the lowest insertion gradient over the entire time horizon $[0, T]$ is within a predefined bound. It should be noted that, solving the optimal mode scheduling problem through this approach still requires many repetitions of Algorithm 2.1. Thus, in the real-time extension of this algorithm, we do not expect the minimum insertion gradient to be below this bound.

It should be noted that, if there are costs associated with individual switchings, i.e. the cost functional contain trajectory cost and switching cost as shown in equation (5) in Chapter 1, then the insertion algorithm has to be modified to account for discontinuous increases in the cost functional associated with mode insertions (see [29]).

An example of the insertion algorithm in action is shown in Figure 10 in order to be compared to Figure 8 because both figures show the solution to optimization problems of the same cost functional (one with fixed number of switchings, one without). Figure 10 shows the resultant switching schedule obtained with the mode insertion technique. Figure 11 shows the cost functional J as a function of the iteration count, which counts the total number of iterations of Algorithm 2.1 (it is executed many times). Mode insertion is done every time when Algorithm 2.1 is completed. It is readily seen that during a run of Algorithm 2.1 for a fixed number of switching times, the cost goes down until it flattens out. Then an insertion is made, which results in a decline in the cost. However, the cost can not be reduced much further when the insertion gradient is close to 0.

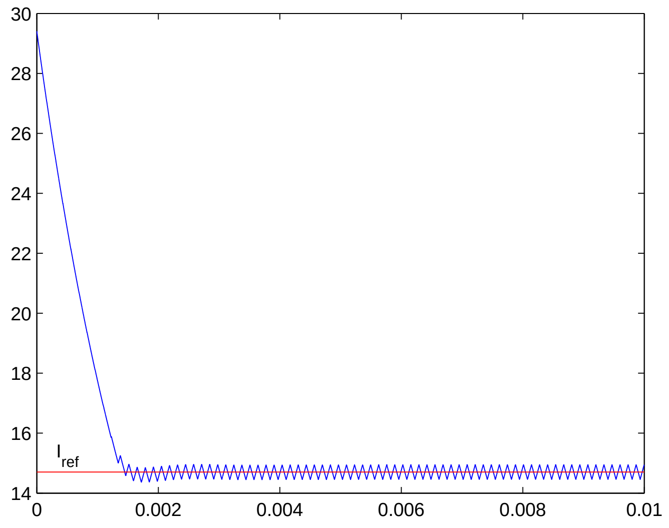


Figure 10: Example of mode scheduling using mode insertion technique. In this example, the optimal control problem is posed to regulate the current of using the circuits shown in Figure 7. In this case, we do not fix the total number of switchings. In this graph, x is the trajectory of the current of the circuit. I_{ref} is the current level that the current regulator circuit aims to track. In this example, the current regulator circuit is initially saturated at around 29A, and the I_{ref} is fixed at around 14.5A. The mode insertion stops after the insertion gradient (14) is below a prescribed bound.

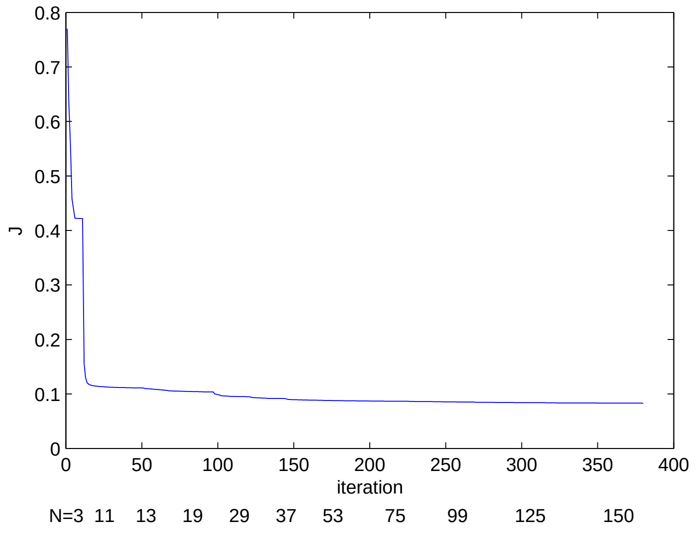


Figure 11: Cost vs. iterations for the off-line mode insertion algorithm executed to get the final trajectory shown in Figure 10. The number of switchings is initially 3, and eventually it grows to 150. This number is shown at the bottom of the figure.

CHAPTER III

OBSERVER-BASED OPTIMAL TIMING CONTROL

In previous chapters, we have discussed several reasons for developing algorithms for real-time optimal control of autonomous switched systems. One important problem is that, for many switched systems in practice, the state of the system can not be measured exactly, or the initial state is unknown, and their states can only be measured via a suitable observer. In this case, a suitable observer has to be constructed for each sub-system, such as the Luenberger observer for linear sub-systems, or the Morall-Grizzle observer [54] for nonlinear sub-systems. Since the open-loop optimal control approaches introduced in the last chapter require the knowledge of the initial state, they are not longer valid and a real-time method to obtain the optimal switching times is needed.

This chapter establishes a real-time optimal control framework (in terms of algorithms) that uses an observer to measure the state of the autonomous switched system. Besides observers, this framework can also be applied to any other state estimation techniques. As such, we do not concern with the details of the state observer for one particular system, but only assume that one exists for the switched system at hand.

Another contributions of this chapter is that we propose a suitable framework to analyze the convergence properties of the proposed algorithms. The need for such a framework is due to the real-time nature of the algorithms, which implies that they can compute only a finite number of iterations during a finite time horizon. Consequently, the usual notion of asymptotic convergence, which is characterized by limits of infinite sequences, cannot be applied. In this case, the convergence analysis is stated in terms of convergence rate. The analysis shows that the performances of the proposed algorithms are characterized by the state estimation error and the time-step of the algorithm, and the analysis can be applied to any estimation techniques. It can even be applied to observers that are not convergent, and we can see the effect of the estimation error on the performances of the proposed algorithms through numerical examples.

This chapter begins by discussing the work of Azuma, Egerstedt and Wardi [11, 37]. They presented the first results on observer-based optimal control of autonomous switched systems by considering the solution point of the optimization problem as a function of the state estimate. Since the state estimate is a function of time, the optimal switching time vector is a function of time as well. The authors then presented a continuous process in the form of a differential equation for the optimal switching time vector. However, this continuous process is not implementable. This chapter then shifts to the original work of this thesis, by extending this approach into an implementable algorithmic framework. Moreover, we provide a framework for the convergence analysis of the proposed algorithms.

3.1 Continuous Process

In [11, 37], the following optimal timing control problem was considered. Given an autonomous switched system in the form of

$$\dot{x} \;=\; f_i(x), \qquad t \in [\tau_{i-1}, \tau_i), \qquad i = 1, \ldots, N+1, \tag{15}$$

where $\{f_i : \mathbb{R}^n \to \mathbb{R}^n\}$ is a collection of functions with the notation $\tau_0 = t$ and $\tau_{N+1} = T$. Suppose that this switched system starts at time t, with initial condition $x(t) = \hat{x}$. The optimal control problem is formulated as

$$\min_{\bar{\tau}} J(t, \hat{x}, \bar{\tau}) = \int_t^T L(x(t))dt, \tag{16}$$

with the switching time vector $\bar{\tau}$ defined as $\bar{\tau} = [\tau_1, \tau_2, ..., \tau_N]^T$. Now, let $\hat{x}$ to be the output of the state observer $\hat{x}(t)$, then one can make the following definition for the optimal switching time process

$$\bar{\tau}^\star(t) = \arg\min_{\bar{\tau}} J(t, \hat{x}(t), \bar{\tau}). \tag{17}$$

From this definition, $\bar{\tau}^\star(t)$ is optimal at time t given the state estimate $\hat{x}(t)$. If the state estimate $\hat{x}(t)$ converges to the "true" state $x(t)$ of the switched system (observer is convergent), then the optimal switching time for the system is obtained if $\bar{\tau}^\star(t)$ is computed. In [11, 37], the authors proposed a method to obtain the evolution $\bar{\tau}^\star(t)$ by a differential equation, so that there is no need to solve Problem (17) for all time t. The differential equation for the switching time evolution of the system was obtained in the following form:

$$\dot{\bar{\tau}}^\star(t) = -\left(\frac{\partial^2 J}{\partial \bar{\tau}^2}(t, x(t), \bar{\tau}^\star(t))\right)^{-1} \left(\frac{\partial^2 J}{\partial t \partial \bar{\tau}}(t, x(t), \bar{\tau}^\star(t)) + \frac{\partial^2 J}{\partial x \partial \bar{\tau}}(t, x(t), \bar{\tau}^\star(t))\dot{x}(t)\right), \tag{18}$$

with the initial condition $\bar{\tau}^\star(0) = \arg\min_{\bar{\tau}} J(0, \hat{x}(0), \bar{\tau})$.

However, differential equation (18) can not be used directly to obtain the optimal switching time evolution. The reason is that, the trajectory $\bar{\tau}^\star(t)$, by definition, is optimal with respect to a time-varying cost functional for all time. Once $\bar{\tau}^\star(t)$ departs from the optimal trajectory (not optimal at one time instant t), there is no guarantee that it will return to it, and the remaining trajectory is no longer optimal. Hence, solving (18) requires the switching times updates to be done in an error free way, which is unfeasible to achieve. In a numerical algorithm point of view, to ensure that $\bar{\tau}^\star(t)$ is obtained as the optimal switching time evolution, the time-step Δt must be 0, and the algorithm requires infinite number of iterations. Hence, this approach is not implementable on any computer hardware. This leads to our contribution, which is extending this approach to an algorithmic framework that can be implemented. Also, note that the goal of the optimal control task is to obtain a local optimal solution for the "true" state of the system, i.e. $\bar{\tau}^\star = \arg\min_{\bar{\tau}} J(0, x(0), \bar{\tau})$. We show how our algorithms update the switching times towards this solution through the analysis of their convergence rates.

3.2 Numerical Algorithms

Consider an autonomous switched system in the form of

$$\dot{x} \ = \ f_i(x), \qquad t \in [\tau_{i-1}, \tau_i), \qquad i = 1, \ldots, N+1, \tag{19}$$

where $\{f_i : \mathbb{R}^n \to \mathbb{R}^n\}$ is a collection of modal functions, with the notation $\tau_0 = 0$, $\tau_{N+1} = T$. $x(t) \in \mathbb{R}^n$ represents the "true" state of the switched system. Suppose that the initial condition $x(0) = x_0$ is not known, and it is estimated by a suitable state observer with the initial condition $\hat{x}_0$. Final time T is given and fixed. For now, the mode sequence $\bar{q} = \{1, 2, ..., N+1\}$ is assumed to be fixed (we consider mode scheduling later in this section).

In order to formulate the problem in a conceptually clean way, it is necessary to reformulate the cost functional of the optimal control problem as a cost-to-go performance functional. First, let $t \in [0, T]$ denote a time point during operation of the system where one step of the proposed algorithm is executed. We denote $q(t)$ to be current mode of operation at time t. we denote the switching times (control variables) of the autonomous switched system at time t as $\bar{\tau}(t) = [\tau_{q(t)}(t), \tau_{q(t)+1}(t), \ldots, \tau_N(t)]^T$. The switching times $\bar{\tau}(t)$ are expected to be updated at each iteration of the algorithm, hence the dependence on t. For simplicity of notation, define $\tau_{q(t)-1} = t$. At time t, we define the the time-varying feasible set

$$\Lambda(t) = \{\bar{\tau}(t) : t = \tau_{q(t)-1}(t) \leq \tau_{q(t)}(t) \leq \ldots, \leq \tau_N(t) \leq \tau_{N+1}(t) = T\}, \tag{20}$$

as the input constraint for the switched system ($\bar{\tau}(t) \in \Lambda(t)$). The real-time framework dictates that, when $t \geq \tau_{q(t)}(t)$, we trigger the mode transition from $q(t)$ to $q(t)+1$, and the dimension of $\bar{\tau}(t)$ is reduced by 1. It should be noted that due to the nature of a real time algorithm, once a switching time $\tau_i(t)$ is surpassed by the time t, it retains its value thereafter.

This switching time vector is assumed to be updated as a discrete process $\{\bar{\tau}(t)\}$

$$\bar{\tau}(t + \Delta t) \;=\; \bar{\tau}(t) + h(t), \tag{21}$$

by a suitable decent vector $h(t)$. Δt is assumed to be the time-step of the proposed algorithms. $h(t)$ is comprised of a step taken during a single iteration of an optimization algorithm.

Now, we are ready to define the cost-to-go functional. We denote this cost-to-go functional $J(t, \hat{x}(t), \bar{\tau}(t))$ to emphasize its dependence on the current system time index t and the state estimator $\hat{x}(t)$. It is defined as follows

$$J(t, \hat{x}(t), \bar{\tau}(t)) \;=\; \int_t^T L(\tilde{x}(s))ds, \tag{22}$$

where $L : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function and $\tilde{x}(s)$ denotes a simulated state trajectory with the initial condition $\tilde{x}(t) = \hat{x}(t)$. The dynamical equation of $\tilde{x}(s)$ is defined as:

$$\begin{aligned} \dot{\tilde{x}}(s) \;&=\; f_i(\tilde{x}(s)), \qquad s \in [\tau_{i-1}(t), \tau_i(t)), i = q(t), \ldots, N+1, \\ &\text{with the initial condition } \tilde{x}(t) = \hat{x}(t). \end{aligned} \tag{23}$$

Note that the dynamical equation of the simulated trajectory is the same as the switched system, but the initial condition of the simulated trajectory is the state estimate $\hat{x}(t)$. Furthermore, note that the simulated state trajectory is indexed by a different time index s to avoid confusion. Hence, at time t, there exist three trajectories that are related to each other. The true state trajectory $x(\xi), \xi \in [0, t]$, is the state of the switched system operated since the initial time $\xi = 0$. The observer output state trajectory $\hat{x}(\xi), \xi \in [0, t]$, is obtained via the observer. The simulated state trajectory obtained at time t, $\tilde{x}(s), s \in [t, T]$, is simulated at each step of the

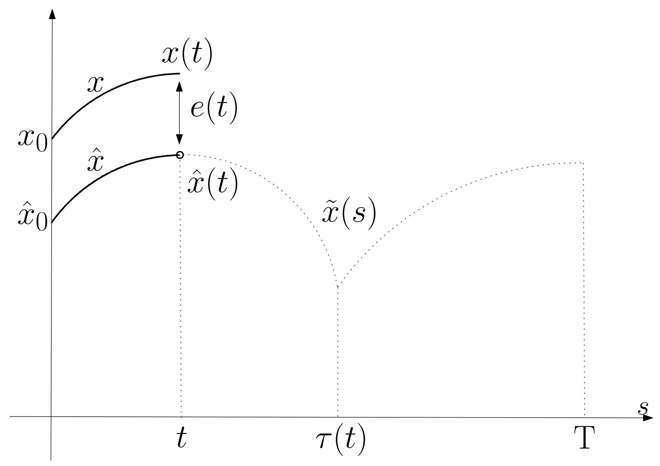


Figure 12: Observer-based on-line timing optimal control. At time t, $x(\cdot)$ is the state trajectory, $\hat{x}(\cdot)$ is the observer output trajectory, and $\tilde{x}(s)$ is future simulated state trajectory indexed by future time index s. The initial condition of $\tilde{x}(s)$ is $\hat{x}(t)$. In this graph, there is only 1 switch, so the control variable is $\tau(t)$.

algorithm. The three trajectories are illustrated in Figure 12. The observer error at time t is defined as $e(t) = \hat{x}(t) - x(t)$.

The eventual goal of the problem is to obtain the optimal switching time with respect to the true state trajectory $x(\cdot)$, and not the observer output $\hat{x}(\cdot)$. In that regard, consider the optimal control problem

$$\min_{\bar{\tau}(t)\in\Lambda(t)} J(t, x(t), \bar{\tau}(t)) \;=\; \int_t^T L(\tilde{x}(s))ds, \tag{24}$$

where $\tilde{x}(s)$ is defined by (23) with the initial condition $\tilde{x}(t) = x(t)$ such as

$$\begin{aligned} \dot{\tilde{x}}(s) \;&=\; f_i(\tilde{x}(s)), \qquad s \in [\tau_{i-1}(t), \tau_i(t)), i = q(t), \ldots, N+1, \\ &\text{with the initial condition } \tilde{x}(t) = x(t). \end{aligned} \tag{25}$$

This problem, which is optimizing the switching time vector at time t for the remaining state trajectory, represents the real-time control task that we wish to achieve. Denote a locally optimal solution to Problem (24) at time t as $\bar{\tau}^\star$. The dimension of $\bar{\tau}^\star$ is the same as $\bar{\tau}(t)$. We should note that, $\bar{\tau}^\star$ is stationary, despite the fact that $J(t, x(t), \bar{\tau}(t))$ depends on time t. This observation is made concrete by the following lemma.

Lemma 3.1 *At time $t = t_0$, if $\bar{\tau}^\star = [\tau^\star_{q(t_0)}, \tau^\star_{q(t_0)+1}, ..., \tau^\star_N]^T$ locally minimizes $J(t_0, x(t_0), \bar{\tau}(t_0))$, then if the system switches according to $\bar{\tau}^\star$ for the remaining state trajectory $x(t), t \in [t_0, T]$, then $[\tau^\star_{q(t)}, \tau^\star_{q(t)+1}, ..., \tau^\star_N]^T$ locally minimizes $J(t, x(t), \bar{\tau}(t))$ for every $t \in [t_0, T]$.*

Proof 3.1 *Since the system switches according to $\bar{\tau}^\star$, then the simulated state trajectory obtained at t_0, $\tilde{x}(s), s \in [t_0, T]$, is identical to the state trajectory $x(t), t \in [t_0, T]$. Then, by Bellman's Optimality Principle (see [13]), $[\tau^\star_{q(t)}, \tau^\star_{q(t)+1}, ..., \tau^\star_N]^T$ locally minimizes $J(t, x(t), \bar{\tau}(t))$ for every $t \in [t_0, T]$.*

$\bar{\tau}^\star$ must satisfy the necessary optimality condition (see [57])

$$\frac{\partial J}{\partial \bar{\tau}}(t, x(t), \bar{\tau}^\star) = 0, \tag{26}$$

and $H = \frac{\partial^2 J}{\partial \bar{\tau}^2}(t, x(t), \bar{\tau}^\star)$ must be positive definite. The algorithms presented in this section compute, at time t, an estimate of the optimal solution $\bar{\tau}^\star$. However, since state information is not known, at each time step of the algorithms, we have to use the cost-to-go functional $J(t, \hat{x}(t), \bar{\tau}(t))$ defined by the state observer $\hat{x}(t)$. In each step of our algorithms, $\frac{\partial J}{\partial \bar{\tau}}(t, \hat{x}(t), \bar{\tau}(t))$ is computed, using gradient formula we discussed in Chapter 2 (13). Hence $\bar{\tau}(t)$ is updated according to information that is available at that time, which is $\hat{x}(t)$, and their dependence on that information defines their real-time nature.

If the system does not switch according to $\bar{\tau}^\star$ at time t, then naturally $\bar{\tau}^\star$ is no longer optimal for the remaining trajectory. In this case, the best the system could do is to switch according to a new locally optimal switching time vector $\arg\min_{\bar{\tau}(t) \in \Lambda(t)} J(t, x(t), \bar{\tau}(t)$. However, the resultant state trajectory generally admits a higher total cost (defined over the entire state trajectory from 0 to T) since the system switched at a non-optimal time. This can be avoided if our algorithms perform good enough so that $\bar{\tau}(t)$ converges to $\bar{\tau}^\star$ before a wrong (premature) switching has occurred (hence if it converges before $t = \tau^\star_1$). This stems from the real-time nature of the problem, and requires our algorithm to allow fast convergence. Furthermore, if a premature switching has occurred, then our algorithm should continue to converge towards the new local optimal, which is the best control action to do at that time. The premature switchings are likely to happen when there are not enough steps taken for the algorithm (large Δt) or there is large observer error. To counter this potential problem, one can perform mode insertion to improve the current mode sequence and restart the algorithms. This aspect will be examined in more detail.

Now we propose two algorithms based on two natural choices for the descent direction $h(t)$, one is based on The Newton-Raphson method and the other is based on gradient descent method.

3.2.1 Newton-Raphson Algorithm

For the Newton-Raphson like method, the Hessian of the cost-to-go functional needs to be computed. The Hessian matrix $H(t)$ is defined by

$$H(t) \;=\; \frac{\partial^2 J}{\partial \bar{\tau}^2}\big(t, \hat{x}(t), \bar{\tau}(t)\big), \tag{27}$$

and assume that $H(t)$ is positive definite. Furthermore, let $\tilde{h}(t)$ be the projection of $-H(t)^{-1}\frac{\partial J}{\partial \bar{\tau}}(t+\Delta t, \hat{x}(t+\Delta t), \bar{\tau}(t))$ onto the feasible set $\Lambda(t)$, and define the step size $\gamma(t)$ by $\gamma(t) \;=\; \max\{\gamma \leq 1 \mid \bar{\tau}(t) + \gamma\tilde{h}(t) \in \Lambda(t)\}$. Finally, we define $h(t)$ by $h(t) = \gamma(t)\tilde{h}(t)$. Note the choice of $\gamma(t)$ ensures that (21) does not overshoot the feasible set and hence the point $\bar{\tau}(t+\Delta t)$ is feasible.

The algorithm is described below.

Algorithm 3.1 *Given time-step Δt:*

Step 1. Obtain the observer output $\hat{x}(t+\Delta t)$. Obtain the simulated state trajectory $\tilde{x}(s), s \in [t+\Delta t, T]$. Compute $\frac{\partial J}{\partial \bar{\tau}}(t+\Delta t, \hat{x}(t+\Delta t), \bar{\tau}(t))$ via the gradient formula (13).

Step 2. Compute $\tilde{h}(t)$, defined as the projection of $-H(t)^{-1}\frac{\partial J}{\partial \bar{\tau}}(t+\Delta t, \hat{x}(t+\Delta t), \bar{\tau}(t))$

onto the feasible set $\Lambda(t)$ at $\bar{\tau}(t)$.

Step 3. Compute $\gamma(t) = \max\{\gamma \leq 1 \mid \bar{\tau}(t) + \gamma\tilde{h}(t) \in \Lambda(t)\}$.

Step 4. Define $h(t) = \gamma(t)\tilde{h}(t)$, and set

$$\bar{\tau}(t + \Delta t) \;=\; \bar{\tau}(t) + h(t). \tag{28}$$

Step 5. Set time $t = t + \Delta t$.

If $t \geq \tau_1(t)$, the system is switched from mode $q(t)$ to $q(t) + 1$. $\bar{\tau}(t)$ is set to $[\tau_{q(t)+1}(t), ..., \tau_N(t)]$. $q(t)$ is increased by 1.

The algorithm stops when all switchings have occurred. If not then let the system evolve until time is at $t + \Delta t$, while computing $H(t)$ during this time. Go back to Step 1 when time is $t + \Delta t$.

The above algorithm starts with an arbitrary condition $\bar{\tau}(0) \in \Lambda(0)$.

To show that this algorithm indeed converges to a local optimal solution $\bar{\tau}^\star$ when a set of conditions are met, the convergence result of this algorithm is addressed next. Asymptotic convergence is a minimal requirement of an algorithm, and typically it means that every accumulation point of an iteration-sequence computed by the algorithm satisfies a suitable optimality condition, like being stationary or a Kuhn-Tucker point. However, Algorithm 3.1 does not compute an infinite sequence of iteration points because $\Delta t > 0$ and $T < \infty$. For this reason we characterize convergence of the algorithm not in an asymptotic sense, but rather in terms of convergence rate.

Consider the Newton-Raphson method for minimizing a twice-continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$:

$$x_{k+1} \;=\; x_k - H(x_k)^{-1}\nabla f(x_k), \qquad k = 1, 2, \ldots, \tag{29}$$

where $H(x) = \frac{d^2 f}{dx^2}(x)$ is the Hessian. Then, if $\lim_{k\to\infty} x_k = x^\star$ for some $x^\star \in \mathbb{R}^n$, and if $H(x^\star)$ is positive definite, then $x^\star$ is a local minimum for f; moreover, there exist $\delta > 0$ and $K > 0$ such that, if $||x_k - x^\star|| < \delta$, then $||x_{k+1} - x^\star|| \leq K||x_k - x^\star||^2$. This means that the algorithm has a quadratic convergence rate (see [57]). Throughout this thesis, $||\cdot||$ denotes the l^2 norm.

The convergence rate that we establish is not quadratic due to the state estimate error $e(t)$ and Δt. In fact, the error term $e(t)$ implies a non-zero offset. Even if $e(t) = 0$, the convergence rate is first-order due to the fact that $\Delta t > 0$. Only if $e(t) = 0$ and $\Delta t \to 0$, will the resulting convergence rate be quadratic.

Theorem 3.1 *Assume that f_i and L are three-times continuously differentiable. Suppose that $\bar{\tau}^\star$ satisfies the necessary optimality condition (26) lying in the interior of the feasible set $\Lambda(t)$, and that $H(t)$ is positive definite. Assume that the system do not undergo mode transition between the time interval $[t, t+\Delta t]$. Then there exist constants $\delta > 0$ and $K > 0$ such that, if $||\bar{\tau}(t) - \bar{\tau}^\star|| < \delta$; $\Delta t < \delta$; $||e(t)|| < \delta$ and $||e(t+\Delta t)|| < \delta$; $\gamma(t) = 1$ in Step 4; and $\tau_1(t) + \Delta t > t$, then*

$$||\bar{\tau}(t+\Delta t) - \bar{\tau}^\star|| \leq \; K\Big(||\bar{\tau}(t) - \bar{\tau}^\star||^2 + \Delta t||\bar{\tau}(t) - \bar{\tau}^\star|| + ||e(t+\Delta t)||\Big). \tag{30}$$

Proof 3.2 *Since $\bar{\tau}^\star$ lies in the interior of the feasible set, there exists $\delta_1 > 0$ such that if $||\bar{\tau}(t) - \bar{\tau}^\star|| < \delta_1$, $\Delta t < \delta_1$, $\gamma(t) = 1$, and switching does not occur at time t, then*

$$\bar{\tau}(t+\Delta t) \;=\; \bar{\tau}(t) - H(t)^{-1}\frac{\partial J}{\partial \bar{\tau}}(t+\Delta t, \hat{x}(t+\Delta t), \bar{\tau}(t)); \tag{31}$$

this is evident by following Steps 1 – 4 of the algorithm 3.1. Hence, we have that

$$\bar{\tau}(t+\Delta t) - \bar{\tau}^\star = \; \bar{\tau}(t) - \bar{\tau}^\star - H(t)^{-1}\frac{\partial J}{\partial \bar{\tau}}\big(t+\Delta t, \hat{x}(t+\Delta t), \bar{\tau}(t)\big). \tag{32}$$

By assumption $\bar{\tau}^\star$ satisfies the necessary optimality condition (26). Therefore, and by Bellman's Principle of Optimality, $\bar{\tau}^\star$ is a local minimum for $J(t+\Delta t, x(t+\Delta t), \bar{\tau})$ as well, and hence

$$\frac{\partial J}{\partial \bar{\tau}}\big(t+\Delta t, x(t+\Delta t), \bar{\tau}^\star\big) \;=\; 0. \tag{33}$$

Consequently, and by (32)

$$\bar{\tau}(t+\Delta t)-\bar{\tau}^\star = \bar{\tau}(t)-\bar{\tau}^\star-H(t)^{-1}\Big(\frac{\partial J}{\partial\bar{\tau}}\big(t+\Delta t,\hat{x}(t+\Delta t),\bar{\tau}(t)\big)-\frac{\partial J}{\partial\bar{\tau}}\big(t+\Delta t,x(t+\Delta t),\bar{\tau}^\star\big)\Big). \tag{34}$$

By the continuity assumption of f_i and L, The function $J(t,\hat{x}(t),\bar{\tau}(t))$ is twice continuously differentiable and its second derivatives are locally Lipschitz continuous. the last difference term in the right-hand side of (34) can be written as

$$\begin{aligned}
&\frac{\partial J}{\partial\bar{\tau}}\big(t+\Delta t,\hat{x}(t+\Delta t),\bar{\tau}(t)\big)-\frac{\partial J}{\partial\bar{\tau}}\big(t+\Delta t,x(t+\Delta t),\bar{\tau}^\star\big)\\
=&\ \frac{\partial^2 J}{\partial\bar{\tau}\partial x}\big(t+\Delta t,x(t+\Delta t),\bar{\tau}^\star\big)e(t+\Delta t)+O\big(||e(t+\Delta t)||^2\big)\\
+&\ \frac{\partial^2 J}{\partial\bar{\tau}^2}\big(t+\Delta t,x(t+\Delta t),\bar{\tau}^\star\big)(\bar{\tau}(t)-\bar{\tau}^\star)+O\big(||\bar{\tau}(t)-\bar{\tau}^\star||^2\big),
\end{aligned}\tag{35}$$

where $\limsup_{\eta\to 0}\frac{O(\eta^2)}{\eta^2}<\infty$*.*

Next, recall that $H(t)=\frac{\partial^2 J}{\partial\bar{\tau}^2}(t,\hat{x}(t),\bar{\tau}(t))$ is positive definite by assumption. Therefore, there exists $\delta_2\in(0,\delta_1)$ and $K_1>0$ such that, if $\Delta t<\delta_2$, $||e(t)||<\delta_2$, and $||\bar{\tau}(t)-\bar{\tau}^\star||<\delta_2$, then

$$||H(t)^{-1}-\Big(\frac{\partial^2 J}{\partial\bar{\tau}^2}\big(t+\Delta t,x(t+\Delta t),\bar{\tau}^\star\big)\Big)^{-1}||\leq\ K_1\Big(\Delta t+||e(t+\Delta t)||+||\bar{\tau}(t)-\bar{\tau}^\star||\Big). \tag{36}$$

Now plug the right-hand side of (35) in (34) to obtain,

$$\begin{aligned}
&\bar{\tau}(t+\Delta t)-\bar{\tau}^\star\\
=&\quad \bar{\tau}(t)-\bar{\tau}^\star-H(t)^{-1}\Big(\frac{\partial^2 J}{\partial\tau\partial x}\big(t+\Delta t,x(t+\Delta t),\bar{\tau}^\star\big)e(t+\Delta t)+O\big(||e(t+\Delta t)||^2\big)\\
&+\ \frac{\partial^2 J}{\partial\bar{\tau}^2}\big(t+\Delta t,x(t+\Delta t),\bar{\tau}^\star\big)(\bar{\tau}(t)-\bar{\tau}^\star)+O\big(||\bar{\tau}(t)-\bar{\tau}^\star||^2\big)\Big).
\end{aligned}\tag{37}$$

Subtracting and adding the term $\left(\frac{\partial^2 J}{\partial \bar\tau^2}(t+\Delta t, x(t+\Delta t), \bar\tau^\star)\right)^{-1}$ *from* $H(t)^{-1}$ *in (37):*

$$\begin{aligned}\bar\tau(t+\Delta t)-\bar\tau^\star \;=\;& \bar\tau(t)-\bar\tau^\star-\Big(H(t)^{-1}-\big(\frac{\partial^2 J}{\partial\bar\tau^2}(t+\Delta t, x(t+\Delta t),\bar\tau^\star)\big)^{-1}\Big)\\ &\Big(\frac{\partial^2 J}{\partial\tau\partial x}\big(t+\Delta t, x(t+\Delta t),\bar\tau^\star\big)e(t+\Delta t)+O\big(\|e(t+\Delta t)\|^2\big)\\ &+\frac{\partial^2 J}{\partial\bar\tau^2}\big(t+\Delta t, x(t+\Delta t),\bar\tau^\star\big)(\bar\tau(t)-\bar\tau^\star)+O\big(\|\bar\tau(t)-\bar\tau^\star\|^2\big)\Big)\\ &-\big(\frac{\partial^2 J}{\partial\bar\tau^2}(t+\Delta t, x(t+\Delta t),\bar\tau^\star)\big)^{-1}\\ &\Big(\frac{\partial^2 J}{\partial\tau\partial x}\big(t+\Delta t, x(t+\Delta t),\bar\tau^\star\big)e(t+\Delta t)+O\big(\|e(t+\Delta t)\|^2\big)\\ &+\frac{\partial^2 J}{\partial\bar\tau^2}\big(t+\Delta t, x(t+\Delta t),\bar\tau^\star\big)(\bar\tau(t)-\bar\tau^\star)+O\big(\|\bar\tau(t)-\bar\tau^\star\|^2\big)\Big), \qquad (38)\end{aligned}$$

and after a bit of algebra, the right-hand side of (38) can be seen to have the form

$$\begin{aligned}&\bar\tau(t+\Delta t)-\bar\tau^\star = -\Big(H(t)^{-1}-\big(\frac{\partial^2 J}{\partial\bar\tau^2}(t+\Delta t, x(t+\Delta t),\bar\tau^\star)\big)^{-1}\Big)\\ &\Big(\frac{\partial^2 J}{\partial\tau\partial x}\big(t+\Delta t, x(t+\Delta t),\bar\tau^\star\big)e(t+\Delta t)+O\big(\|e(t+\Delta t)\|^2\big)\\ &+\frac{\partial^2 J}{\partial\bar\tau^2}\big(t+\Delta t, x(t+\Delta t),\bar\tau^\star\big)(\bar\tau(t)-\bar\tau^\star)+O\big(\|\bar\tau(t)-\bar\tau^\star\|^2\big)\Big)\\ &-\big(\frac{\partial^2 J}{\partial\bar\tau^2}(t+\Delta t, x(t+\Delta t),\bar\tau^\star)\big)^{-1}\\ &\times\Big(\frac{\partial^2 J}{\partial\tau\partial x}\big(t+\Delta t, x(t+\Delta t),\bar\tau^\star\big)e(t+\Delta t)+O\big(\|e(t+\Delta t)\|^2\big)+O\big(\|\bar\tau(t)-\bar\tau^\star\|^2\big)\Big).\end{aligned} \qquad (39)$$

Consequently, using (39) and (36), there exist $K_2>0$ *and* $\delta_3\in(0,\delta_2)$ *such that, if* $\Delta t<\delta_3$, $\|e(t)\|<\delta_3$, $\|e(t+\Delta t)\|<\delta_3$, *and* $\|\bar\tau(t)-\bar\tau^\star\|<\delta_3$, *then*

$$\begin{aligned}\|\bar\tau(t+\Delta t)-\bar\tau^\star\| <\; & K_2\Big(\Delta t+\|e(t+\Delta t)\|+\|\bar\tau(t)-\bar\tau^\star\|\Big)\Big(\|e(t+\Delta t)\|\\ &+\|\bar\tau(t)-\bar\tau^\star\|\Big)+\;K_2\Big(\|e(t+\Delta t)\|+\|\bar\tau(t)-\bar\tau^\star\|^2\Big).\end{aligned} \qquad (40)$$

Finally, by reducing K_2 *and* δ_2 *if necessary, there exist* $K>0$ *and* $\delta>0$ *such that, if the assumptions of the lemma are satisfied, then Equation (30) is in force.*

It is important to make an observation for Algorithm 3.1, and explain why we use the descent direction $-H(t)^{-1}\frac{\partial J}{\partial\bar\tau}(t+\Delta t,\hat x(t+\Delta t),\bar\tau(t))$. The reason is that, when

$\Delta t \to 0$, and if the current switching time $\bar{\tau}(t)$ is optimal with respect to the state observer output $\hat{x}(t)$, Algorithm 3.1 becomes the same as the continuous process outlined in the previous section. Namely, we can derive the following equation by using Taylor Expansion on equation (18)

$$\begin{aligned}
\dot{\bar{\tau}}^\star(t) &= -\left(\frac{\partial^2 J}{\partial \tau^2}(t, x(t), \bar{\tau}^\star(t))\right)^{-1} \left(\frac{\partial^2 J}{\partial t \partial \bar{\tau}}(t, x(t), \bar{\tau}^\star(t)) + \frac{\partial^2 J}{\partial x \partial \bar{\tau}}(t, x(t), \bar{\tau}^\star(t))\dot{x}(t)\right) \\
&= -\left(\frac{\partial^2 J}{\partial \tau^2}(t, x(t), \bar{\tau}^\star(t))\right)^{-1} \left(\lim_{\Delta t \to 0} \frac{1}{\Delta t} \frac{\partial J}{\partial \bar{\tau}}(t + \Delta t, \hat{x}(t + \Delta t), \bar{\tau}^\star(t))\right). \qquad (41)
\end{aligned}$$

This means that, if $\bar{\tau}(t)$ is optimal with respect to the state observer output $\hat{x}(t)$, and Δt is very small, then Algorithm 3.1 *preserves* optimality in a sense that $\bar{\tau}(t + \Delta t) = \bar{\tau}(t) - H(t)^{-1}\frac{\partial J}{\partial \bar{\tau}}(t + \Delta t, \hat{x}(t + \Delta t), \bar{\tau}(t))$ is a very good approximation as a local optimal with respect to the state observer output $\hat{x}(t + \Delta t)$, and is exactly optimal as $\Delta t \to 0$. This property is unique to Algorithm 3.1, and is absent in the gradient descent algorithm which we will introduce in the next section.

We should also make an observation on the computation load.. At each iteration time $t + \Delta t$, when the observer estimate $\hat{x}(t + \Delta t)$ is available, the Hessian matrix $H(t)$ is assumed to be already computed, and the computation required is the forward simulation for the gradient $\frac{\partial J}{\partial \bar{\tau}}(t + \Delta t, \hat{x}(t + \Delta t), \bar{\tau}(t))$. After the switching times are updated, between time $t + \Delta t$ to $t + 2\Delta t$, the Hessian matrix $H(t + \Delta t)$ is computed. Hence, this algorithm balances computation load quite well. This is an important feature, since not all computations for the algorithm is performed at the time when updated information is available. This feature is desirable for a real-time algorithm.

However, there are two drawbacks for this algorithm. First, like all Newton-like algorithms, if current iterate of $\bar{\tau}(t)$ is not sufficiently close to the optimal solution $\bar{\tau}^\star$, then the algorithm may not converge or even progress towards $\bar{\tau}^\star$. Second, the Hessian $H(t)$ may be ill-conditioned or singular and thus not positive definite. These two drawbacks can be avoided in the slower converging but globally stable gradient descent-based algorithm. Before we move on, let us first examine the effects of the convergence analysis through a numerical example.

3.2.2 Numerical Example to Illustrate Convergence Properties

As we can see from (30), the time-step Δt and the observer plays an important role in convergence property of the algorithm.

Consider the two-dimensional, linear system defined by $\dot{x} = A_i x$, $y = c_i x$, $i = 1, 2$, where

$$A_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix}, \quad \text{and } c_1 = c_2 = \begin{bmatrix} 1, & 1 \end{bmatrix}^T,$$

and let the cost functional be $J = \int_0^T ||x(t)||^2 dt$. We use the Luenberger observer so that the equation for the state estimator is $\dot{\hat{x}} = A_i \hat{x} - \ell_i c_i (x - \hat{x})$. The initial conditions are $x_0 = [0.55, 0.55]^T$ and $\hat{x}_0 = [0.3, 0.7]^T$, and we choose $T = 5$. We start at mode 1, namely $\dot{x} = A_1 x$, and consider a single switching to mode 2 at a time $\tau \in [0, T]$, where τ serves as the variable for the optimization problem (we drop the vector notation since τ is a scalar). The initial value for τ, where the algorithm starts, is at $\tau = 2.5$, and the optimal point, denoted by $\tau^\star$, computed analytically, is $\tau^\star = 1.71$.

We made four runs of Algorithm 3.1, with two values of Δt and two sets of observer matrices ℓ_i, and the results are shown in Figure 13. In part A of the figure, $\Delta t = 0.05$, and the observer matrices are $\ell_1 = [-1, 5]^T$ and $\ell_2 = [3.333, 0.667]^T$ so that the eigenvalues of the observer subsystems are at $-2 \pm j$ for both $i = 1$ and $i = 2$. The graph in the figure is of the variable τ computed by the algorithm, as a function of time t, while the horizontal line indicates the optimal value $\tau = \tau^\star = 1.71$. The dashed diagonal line is the graph of t, and the vertical line indicates the time t where $\tau = t$. Prior to that time $\tau > t$, namely the switching time is in the future (with respect to t) and hence τ can be modified by the algorithm. After that time $\tau < t$, the switching has already occurred and hence τ cannot be changed by the algorithm.

This is apparent from the figure, where τ approaches the value of (about) 1.6, and it remains a constant once the switching has occurred and become part of the past.

The algorithm's convergence can be sped up by reducing Δt or choosing a more aggressive observer. First, we kept Δt at 0.05 (as in part A) and changed the observer matrices t, $\ell_1 = [-37, 53]^T$ and $\ell_2 = [35.333, -19.333]^T$, so that the eigenvalues of the matrices $A_i - \ell_i c_i$ are at $-8 \pm 5j$ for both $i = 1$ and $i = 2$. The results are shown in part B of the figure, and they indicate better convergence than in part A. Next, we reduced Δt from 0.05 to 0.005 and used the same observer as in part A, and the results, shown in part C of the figure, also indicate an improvement over those in part A. For best results, shown in part D, we used the observer of part B and Δt of part C.

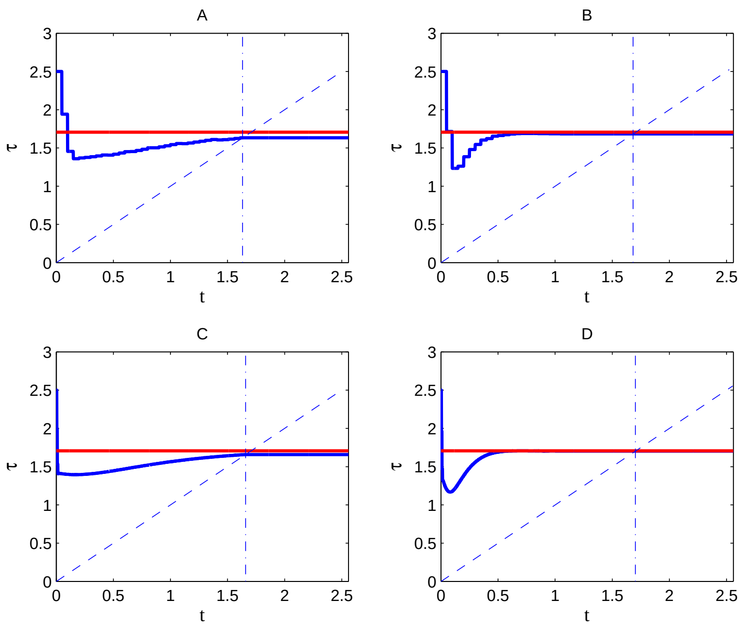


Figure 13: Real time optimal control of autonomous switched system using the Newton-Raphson Algorithm for one switching case. The switching time evolution $\tau(t)$ as a function of time with different Δt and observers are plotted in four graphs for comparison to show the effects of these parameters on the convergence of the algorithm. The red solid straight line is the optimal switching time for the system, and the blue solid curve is the switching time evolution $\tau(t)$.

It is evident from Figure 13 that the use of a more aggressive observer yields faster convergence of the algorithm. Less pronounced but also evident is the fact that smaller Δt yields faster convergence. To further test this point, we set the estimation error to 0 (by setting $\hat{x}(0) = x(0)$), and ran the algorithm with three values of Δt: 0.1, 0.01, and 0.001. Starting at $\tau(0) = 2.5$, Table 1 shows the quantities $\ln(|\tau_k - \tau^\star|)$ for the first 7 iterations of the algorithm and for all three values of Δt, where τ_k is the switching time at the start of the *kth* iteration. Recall that Δt is not the integration interval (in all three runs the integration interval was 0.001), but the time between two consecutive iterations of the algorithm. As mentioned earlier, if the estimation error can be neglected, then with a small Δt the right-hand side of (30) would be dominated by the term $||\bar{\tau}(t) - \bar{\tau}||^2$ and hence indicate a quadratic convergence rate, while if Δt is large then the right-hand side of (30) would be dominated by $\Delta t||\bar{\tau}(t) - \bar{\tau}||$, indicating a linear convergence rate. The results of Table 1 clearly show faster convergence for smaller Δt.

k	0	1	2	3	4	5	6
$\Delta t = 0.1$	-0.2307	-1.4002	-2.4229	-2.8109	-2.9854	-3.0477	-3.0679
$\Delta t = 0.01$	-0.2307	-1.2050	-3.9477	-5.4536	-5.1579	-5.0772	-5.7839
$\Delta t = 0.001$	-0.2307	-1.1899	-4.3522	-7.6708	-8.6631	-9.4915	-10.0498

Table 1: $\ln(|\tau_k - \tau^\star|)$ for $k = 0, \ldots, 6$

3.2.3 Gradient Descent Algorithm

The algorithm presented in this sub-section is based on the process $\{\bar{\tau}(t)\}$ as defined in (21) using the gradient descent direction. The algorithm has the following form.

Algorithm 3.2 *Given time-step Δt:*

Step 1. Obtain the observer output $\hat{x}(t + \Delta t)$. Obtain the simulated state trajectory $\tilde{x}(s), s \in [t + \Delta t, T]$. Compute $\frac{\partial J}{\partial \bar{\tau}}(t + \Delta t, \hat{x}(t + \Delta t), \bar{\tau}(t))$ via the gradient formula (13).

Step 2. Compute $\tilde{h}(t)$, defined as the projection of $-\frac{\partial J}{\partial \bar{\tau}}(t + \Delta t, \hat{x}(t + \Delta t), \bar{\tau}(t))$ onto *the feasible set $\Lambda(t)$ at $\bar{\tau}(t)$.*

Step 3. Compute $\gamma(t) = \max\{\gamma \leq 1 \mid \bar{\tau}(t) + \gamma\tilde{h}(t) \in \Lambda(t)\}$.

Step 4. Define $h(t) = \gamma(t)\tilde{h}(t)$, and set

$$\bar{\tau}(t + \Delta t) \;=\; \bar{\tau}(t) + h(t). \tag{42}$$

Step 5. Set time $t = t + \Delta t$.

If $t \geq \tau_1(t)$, the system is switched from mode $q(t)$ to $q(t) + 1$. $\bar{\tau}(t)$ is set to $[\tau_{q(t)+1}(t), ..., \tau_N(t)]$. $q(t)$ is increased by 1.

The algorithm stops when all switchings have occurred. If not then let the system evolve until time is at $t + \Delta t$, while computing $H(t)$ during this time. Go back to Step 1 when time is $t + \Delta t$.

The above algorithm starts with an arbitrary condition $\bar{\tau}(0) \in \Lambda(0)$.

This algorithm utilizes gradient decent direction as opposed to Newton-Raphson decent direction. Such algorithms are known to converge globally to Kuhn-Tucker points. They also have linear convergence rate. The convergence result for Algorithm 3.2 is again based on the notion of linear convergence, and it is described in the next theorem.

Theorem 3.2 *Suppose that $\bar{\tau}^\star$ satisfies the necessary optimality condition (26) lying in the interior of the feasible set $\Lambda(t)$, and that the Hessian $H = \frac{\partial^2 J}{\partial \bar{\tau}^2}\big(t, x(t), \bar{\tau}^\star\big)$ is positive definite. Assume that the system do not undergo mode transition between the time interval $[t, t+\Delta t]$. Then there exist constants $\delta > 0$, $K_1 \in (0, 1)$, $K_2 > 0$, $\eta_1 > 0$, and $\eta_2 > \eta_1$, such that, if $||\bar{\tau}(t) - \bar{\tau}^\star|| < \delta$; $\Delta t < \delta$; $||e(t)|| < \delta$ and $||e(t + \Delta t)|| < \delta$, $\tau_1(t) + \Delta t > t$, and if $\eta_1 < \gamma(t) < \eta_2$, then*

$$||\bar{\tau}(t + \Delta t) - \bar{\tau}^\star|| \leq K_1||\bar{\tau}(t) - \bar{\tau}^\star|| + K_2\eta_2||e(t + \Delta t)||. \tag{43}$$

Proof 3.3 *Since $\bar{\tau}$ lies in the interior of the feasible set, there exists $\delta_1 > 0$ such* *that, if* $||\bar{\tau}(t) - \bar{\tau}^\star|| < \delta_1$*, then*

$$\bar{\tau}(t+\Delta t) = \bar{\tau}(t) - \gamma(t)\frac{\partial J}{\partial \bar{\tau}}(t+\Delta t, \hat{x}(t+\Delta t), \bar{\tau}(t)). \tag{44}$$

Since $\bar{\tau}^\star$ *is a stationary point for the switching time optimization problem, we have that* $\frac{\partial J}{\partial \bar{\tau}}(t, x(t), \bar{\tau}^\star) = 0$*, and moreover Bellman's Principle of Optimality Principle,* $\bar{\tau}^\star$ *is also a local minimum for* $J(t+\Delta t, x(t+\Delta t), \bar{\tau}^\star)$*, and hence*

$$\frac{\partial J}{\partial \bar{\tau}}\big(t+\Delta t, x(t+\Delta t), \bar{\tau}^\star\big) \;=\; 0. \tag{45}$$

By the continuity assumption of f_i *and* L*, The function* $J(t, \hat{x}(t), \bar{\tau}(t))$ *is twice continuously differentiable and its second derivatives are locally Lipschitz continuous. Thus, we have the following equation*

$$\begin{aligned}
& \bar{\tau}(t+\Delta t) - \bar{\tau}^\star = \bar{\tau}(t) - \bar{\tau}^\star - \gamma(t)\frac{\partial J}{\partial \bar{\tau}}(t+\Delta t, \hat{x}(t+\Delta t), \bar{\tau}(t)) \\
= \quad & \bar{\tau}(t) - \bar{\tau}^\star - \gamma(t)\Big(\frac{\partial J}{\partial \tau}(t+\Delta t, \hat{x}(t+\Delta t), \bar{\tau}(t)) - \frac{\partial J}{\partial \bar{\tau}}(t+\Delta t, x(t+\Delta t), \bar{\tau}^\star)\Big) \\
= \quad & \bar{\tau}(t) - \bar{\tau}^\star - \gamma(t)\Bigg(\frac{\partial^2 J}{\partial \bar{\tau}\partial x}(t+\Delta t, x(t+\Delta t), \bar{\tau}^\star)e(t+\Delta t) + O(||e(t+\Delta t)||^2) \\
& + \frac{\partial^2 J}{\partial \tau^2}(t+\Delta t, x(t+\Delta t), \bar{\tau}^\star)(\bar{\tau}(t) - \bar{\tau}^\star) + O(||\bar{\tau}(t) - \bar{\tau}^\star||^2)\Bigg) \\
= \quad & \Big(I - \gamma(t)\frac{\partial^2 J}{\partial \tau^2}(t+\Delta t, x(t+\Delta t), \bar{\tau}^\star)\Big)(\bar{\tau}(t) - \bar{\tau}^\star) \\
& -\gamma(t)\frac{\partial^2 J}{\partial \bar{\tau}\partial x}(t+\Delta t, x(t+\Delta t), \bar{\tau}^\star)e(t+\Delta t) \\
& -\gamma(t)\Big(O(||\bar{\tau}(t) - \bar{\tau}^\star||^2) + O(||e(t+\Delta t)||^2)\Big),
\end{aligned} \tag{46}$$

where I *is the identity matrix whose dimension is that of the Hessian* H*. Next, let* λ_{max} *and* λ_{min} *denote the largest eigenvalue and the smallest eigenvalue of* H*, respectively. Fix* $\eta_2 > 0$ *such that* $1 - \eta_2\lambda_{max} > \frac{1}{2}\eta_2\lambda_{max}$*, and fix* $\eta_1 \in (0, \eta_2)$*. For every* $\gamma(t) \in (\eta_1, \eta_2)$*, the matrix* $I - \gamma(t)H$ *is symmetric and positive definite, and its matrix norm satisfies the inequality*

$$||I - \gamma(t)H|| \;\leq\; 1 - \eta_1\lambda_{min}. \tag{47}$$

Fix $K_1 \in (1-\eta_1\lambda_{min}, 1)$. By (46), there exist constants $\delta \in (0,\delta_1)$ and $K_2 > 0$ such that, if the conditions of the proposition's assertion are satisfied, then Equation (43) is in force. This completes the proof.

From the convergence properties of both algorithms presented, it is clear that the time-step Δt and the observer error $e(t)$ dictate the performance of the algorithms. In fact, if the observer error is large, none of the proposed algorithm guarantees to converge to the optimal solution $\bar{\tau}^\star$ due to the offset caused by $e(t)$ in the convergence analysis. This is to be expected, since the state information is most important in determining the optimal control.

3.2.4 Combined Algorithm and On-line Mode Scheduling

Comparing the convergence analysis (30) of Algorithm 3.1 and the one (43) for Algorithm 3.2, one can see that these two algorithms can be used in conjunction. Specifically, one can use gradient descent version (Algorithm 3.2) at beginning, until the gradient $\frac{\partial J}{\partial \bar{\tau}}(t+\Delta t, \hat{x}(t+\Delta t), \bar{\tau}(t))$ is below a threshold, then switch to Newton-Raphson version (Algorithm 3.1). This combined algorithm achieves better convergence result for a few reasons. First, at the beginning of operation of the switched system, the initial guess of switching times may not be close to the optimal solution. In this case, the Hessian matrix may be ill conditioned. Furthermore, Algorithm 3.2 guarantees to converge towards $\bar{\tau}^\star$, provided that the state estimate error $e(t)$ is 0. The Newton-Raphson version does not have this property. Lastly, Algorithm 3.2 requires significantly less computation (Hessian matrix is not needed) than Algorithm 3.1, but it has slower convergence speed (in terms of convergence rate). In the other hand, due to the special property we pointed out in Section 3.2.1, the Newton-Raphson algorithm should be used when the gradient $\frac{\partial J}{\partial \bar{\tau}}(t, \hat{x}(t), \bar{\tau}(t))$ is close to zero to maintain the optimality of the switching time evolution.

Due to observer error or higher Δt, it is possible that a switching time "freezes" before it converges to the optimal value. This problem can be alleviated by the operations of mode insertion defined in Chapter 2. After all switchings occurred at time t, insertion gradient can be computed using equation (14) over a time-grid $[t, T]$, and the lowest insertion point are selected and an mode insertion operation can be performed. We emphasize that we are in no way trying to obtain a optimal mode sequence, as optimal mode scheduling problem is in general infeasible to be solved on-line. However, we do on-line mode scheduling by mode insertions if the cost functional goes down when a mode insertion is made. Note that, after a mode insertion, the convergence process of the algorithm starts afresh, and possibly under the condition that the state estimation process has had some time to settle closer to the state variable. This will be demonstrated by a numerical example in the next sub-section.

Finally, it should be noted that the algorithms developed in this chapter apply perfectly well to the case where state information is known exactly. In this case, the convergence analysis is the same except $e(t) = 0$ for all time t. Hence, the results presented in this chapter may be useful for applications in which the observers are not needed, but on-line computations of the optimal switching times are required.

3.2.5 Tracking Example with Mode Insertion

To illustrate the combined algorithm and how mode insertions improve the overall cost, we look at another numerical example, this time involving tracking. Consider the two-dimensional system having the modes $\dot{x} = Ax$ or $\dot{x} = Ax + b$ (denoted by Mode 1 and Mode 2, respectively), where

$$A = \begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix}$$

and $b = [1, 1]^T$. The problem under study is to track the linear curve $r(t), t \in [0, 10]$, defined by $r(t) = [0.1t, 0.025t]^T$, by the state trajectory $x(\cdot)$. Thus, we attempt to minimize the cost functional $J = \frac{1}{2}\int_0^T ||x(t) - r(t)||^2 dt$, with $T = 10$, as a function of the switching times. The number of the switchings is not fixed a priori, and it is modified in the manner described below. The system starts at Mode 1 and then alternates between the two modes. The output equation of the system is $y = cx$ with $c = [1, 1]$, and the (Luenberger) observer matrix is $\ell = [0.5, -0.5]^T$, so that the eigenvalues of the observer's system matrix $A - \ell c$ are both at -1. The initial state is $x(0) = [0.55, 0.55]^T$, and the initial state estimator is $\hat{x}(0) = [0, 2]^T$.

It is not unlikely that the optimal tracking would involve a sliding-mode switching control policy requiring infinite switchings between Mode 1 and Mode 2. However, as is the case in many applications, the number of switchings has to be limited for practical reasons, and therefore we restrict it, somewhat arbitrarily, to 5. Thus, starting at a pre-specified initial point $\bar{\tau}(0)$, the algorithm computes iteration-points $\bar{\tau}(t) = [\tau_1(t), \ldots, \tau_5(t)]^T$ at times $k\Delta t$, $k = 1, 2, \ldots$. However, if at some point $t_1 < T$ it happens that $\tau_1(t_1) \leq t_1$, then ahead of this point the first switching time is a part of the past and cannot be modified; we say that the switching time τ_1 freezes and it is no longer a part of the variable parameter whose dimension is reduced to 4. If τ_2 later freezes at a future time $t_2 \in (t_1, T]$, then the variable parameter becomes 3-dimensional. Continuing in this way, all of the switching times may freeze, and this can happen prematurely before the algorithm computes an adequate approximation to the optimal schedule.

To get around this problem we insert two switching times when the dimension of the variable becomes 3. It is done in the following way. Given a time $t \in (0, T)$ which is not a switching time, consider inserting the complementary mode to the mode that is active at time t, over the interval $[t - \lambda, t + \lambda)$, for some $\lambda > 0$. If λ is small enough then this insertion results in adding two switching times, at the two end-points of the interval, to the schedule. Denoting the cost functional J as a function of λ by $\tilde{J}(\lambda)$, let us define $D(t) = \frac{d\tilde{J}}{d\lambda^+}(0, t)$, where $\frac{d\tilde{J}}{d\lambda^+}(0, t)$ is computed via the insertion gradient formula (14). Thus, if $D(t) < 0$ then such an insertion at a small-enough interval would reduce the value of the cost functional J. We choose the point t to be the point τ where $D(\tau)$ obtained its minimum value over a suitable grid of the interval $[0, T]$, as long as this minimum was negative (which it was, in our case). We then inserted the complementary mode at time t for an interval of duration $\lambda = 0$, namely both new switching times are at t, and the fact that $D(t) < 0$ ensures that the algorithm will act to separate them.

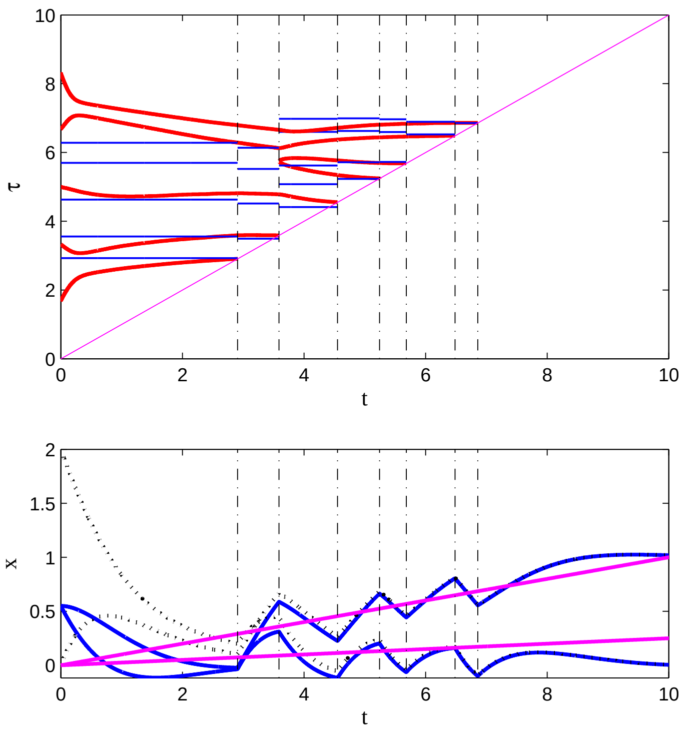


Figure 14: Real-time optimal control of autonomous switched system using gradient descent Algorithm first, then switch to Newton-Raphson method. The top graph plots the switching times $\tau(t)$ vs. t, where the red solid curve is the switching time evolution $\tau(t)$ and blue solid straight lines indicate the optimal switching times based on perfect state information, computed initially and every time when switchings occurred. The bottom graph plots the state (in blue solid curves) and observer (in black dotted curves) trajectories, as well as the curve set to be tracked (in purple solid straight lines).

The results of the algorithm run are shown in Figure 2. The upper graph shows the evolution of the vector $\bar{\tau}(t)$ as a function of time t. Starting at

$\bar{\tau}(0) = [1.7, 3.3, 5.0, 6.7, 8.3]^T$, the switching times are drawn by the thick curves. The piecewise-horizontal lines show the optimal switching times for a given schedule, computed off line as points of reference. τ_1 freezes at time $t_1 = 2.93$; at this time the problem is changed, its dimension reduced to 4, and its solution point and the horizontal lines that mark it are changed as well. τ_2 later freezes at time $t_2 = 3.59$, and at this point we perform a mode-insertion at the time $t = 5.75$. Now the number of variables is restored to 5, and the algorithm continues its course until the switching times freeze one-at-a-time, and this happens at the time-points 4.55, 5.24, 5.68, 6.49, and 6.86. The graph indicates that these points are close to the optimal points as indicated by the horizontal lines, and therefore we do not insert any more new modes. Note that computing the optimal solution (marked by the horizontal lines) is done only to gauge the algorithm's performance. The lower graph of the figure shows the evolution of the two components of the state trajectory (thick curves), the two components of the state estimator (dashed curves), and the two co-ordinates of the line that has to be tracked (diagonal lines). We clearly discern a trend toward improving the tracking objective.

3.3 Conclusions

This chapter presents an algorithmic framework for observer-based optimal control of autonomous switched systems. The need for on-line algorithms arises in situations where the state variable cannot be measured, and it is estimated by a suitable state observer. However, the framework is applicable to a broader setting with any state estimator. Each iteration of the proposed algorithms attempts to reduce the cost-to-go based on the value of the state estimator at that time. Once a switching time is reached by the (real) computing time, it moves into the past can not be modified by the algorithm. If the algorithm has not converged at that time, then the system switches prematurely. This can be cause by large step-size Δt and observer error. To counter this, we propose to improve the mode sequence on-line using mode insertion.

The main contribution of this chapter are the proposed algorithms and characterizations of the algorithms' convergence behaviors via upper bounds on their convergence rates. Two specific algorithms are presented: a Newton-Raphson based algorithm, and a gradient descent based algorithm. For these algorithms we derive respective convergence-rate bounds in terms of the distance of a computed switching time sequence from an optimal one, the time-step Δt, and the estimation error. The derived results extend well-known results on convergence rates from the standard setting of nonlinear programming to the present setting of real-time optimization of switching systems. Numerical examples support the theoretical results.

CHAPTER IV

ROBUST OPTIMAL TIMING CONTROL OF SWITCHED SYSTEMS

This chapter addresses the problem of designing a state-feedback optimal controller for autonomous switched systems that are subject to noise and disturbances. Hence, the goal of this chapter is to present a real-time optimal control algorithm which is robust under disturbances. Throughout this chapter, we assume that the mode sequence $\bar{q}$ is fixed.

We should first note that, the real-time control framework established in Chapter 3, although designed for an observer, can also be used to cope with disturbances that have large amplitude but low frequency, since we can consider the state measurement error as an one-time disturbance of a large amplitude. However, due to time required for computation, when the system is operating under a noisy environment where disturbances have low amplitude but high frequency, corrections for deviations from the optimal trajectory with this framework may not feasible.

In this chapter, we propose a robust controller that incorporates a new approach into the previous framework of feedback control of switched systems, by using the optimal switching planes located at the optimal switching times. We use the state update to check for intersection of the state trajectory with the switching plane. These intersections trigger mode transitions, and their detections require very little computational effort, since no forward simulation of the state trajectory is necessary. When the switching planes are optimal, they implement a neighboring optimal control scheme, which is capable of compensating small amplitude deviations from the optimal path. Slowly-varying deviations of large amplitude, on the other hand, are taken care of by real-time adjustments of the planes.

An important contribution of this chapter is the identification of the class of problems for which this optimal switching plane controller is applicable. We show in this chapter that, a stationary optimal switching surface and its tangential approximation (an optimal switching plane) exists for individual switching times, only if the optimal control problem at hand is a free final-time problem. Hence, the switching plane controller proposed in this chapter cannot be used for fixed final-time problems. We show in this chapter that, if applicable, the proposed optimal plane controller is more robust in a noisy environment than the real-time frameworks established in Chapter 3.

We begin the discussion with our definitions of implicit and explicit state-feedback control and state their differences. Then, we establish a result on stationary switching manifolds, which are necessary to justify the optimal switching plane controller. Finally, we describe our control approach, which unifies both state-feedback approaches, and present the algorithm, as well as its performance analysis, that supports this framework.

4.1 Implicit and Explicit State-feedback Control

In the past literature, there have been a number of approaches in optimal control of hybrid systems, that use switching manifolds as *event generators*. They generate a switching event when the state of the hybrid system $x(t)$ satisfies a switching inequality condition $\Phi(x(t)) \leq 0$ or a switching equality condition $\Phi(x(t)) = 0$, defined by an event function $\Phi(x(t))$. We call this control approach as an *explicit* state-feedback control. With an explicit state-feedback control, the switching events can be checked reliably with negligible computational effort. Example of explicit state-feedback control is a hybrid system controlled by a switching manifold (mode transitions are triggered when the state intersects with this manifold), which is defined as a compact set $\{x : \Phi(x) = 0\}$.

Note that, based on the definition of an autonomous switching hybrid system (ASHS), switching transitions in an ASHS are generated by switching manifolds. In the case of ASHS, the switching manifolds are fixed a priori or parametrized by a set of parameters. In both cases, the switching manifolds are not computed or updated on-line, and they only serve to provide switching events for the hybrid system. Thus, the optimal control problems considered in the past literature using this control approach are either to optimize the state trajectory over an external control input (see [62, 63]) or to optimize over the parameter values (see [9, 18]). As these manifolds are designed with respect to one specific optimal trajectory, the system is generally not optimal in the presence of disturbances.

Instead of pre-determined switching manifolds, we are interested in the *stationary optimal switching manifolds* of a switched system. The switching manifolds are optimal in a region of state-space if all switchings on the manifolds correspond to optimal switchings of an optimal control problem. Hence, this explicit state-feedback controller, if exists, is very robust since the switching events generated by the optimal switching manifolds are always optimal. However, the optimal switching manifolds, being a set of states at the optimal switching times, are typically computational infeasible to be obtained on-line. We instead aim for the tangential approximation of these surfaces, which we call the *optimal switching planes.* We shall see that the optimal switching planes implement a neighboring optimal control scheme, which is capable of compensating small amplitude deviations from the nominal path. Slowly-varying deviations of large amplitude, on the other hand, are taken care of by real-time adjustments of the planes.

Stationarity of the optimal switching manifolds is important (otherwise they no longer comprise a state-feedback controller), and we shall see that stationarity holds if the corresponding optimal control problem is a free final-time problem.

We consider the real-time framework established in Chapter 3 as an example of *implicit state-feedback control*, for which at each time-step, one iteration of an optimization algorithm is executed using state measurements. In this chapter, we merge implicit and explicit feedback control into a unified framework. As a core feature, the state-dependent switching policy is expressed in terms of time-varying switching planes, which triggers mode transitions when being intersected by the state trajectory. These switching planes are updated in real time using the state measurements.

4.2 Stationarity of Locally Optimal Switching Planes

In this section we show the existence of the stationary optimal switching manifolds for an optimal control problem with no constraint on the final time. To capture the salient point of the argument, we show this property for a bi-modal, single-switch system of the form

$$\dot{x} = \begin{cases} f_0(x), & t \in [0, \tau) \\ f_1(x), & t \in [\tau, T] \end{cases}, \tag{48}$$

with smooth modal functions $f_i : \mathbb{R}^n \to \mathbb{R}^n$. $x(t) \in \mathbb{R}^n$ is the state variable. The initial condition $x(0) = x_0$ is assumed to belong to a set $\mathcal{X}_0$. We assume that the final time T is not fixed and it is part of the control variable $\bar{\tau} = [\tau, T]^T$ with the constraint $0 \leq \tau \leq T$. We denote $T = \tau_2$ and $0 = \tau_0$ for simplicity of notation. We denote $x_k = x(\tau_k)$ and they are called *switch points*. We introduce the notion of *activation duration* $\delta_0 = \tau$, and $\delta_1 = T - \tau$. The time span δ_k represents the duration of activation for mode k. We call the sequence $\bar{\delta} = [\delta_0, \delta_1]^T$ the *activation duration sequence*. Note that the activation duration vector $\bar{\delta} = [\delta_0, \delta_1]^T$ is completely equivalent (related with a simple equivalence relation) to the switching time vector $\bar{\tau} = [\tau, T]^T$, and either can be used as the control variable. We denote the state trajectory under the dynamics (48) with initial condition x_0, activation duration $\bar{\delta}$ as $x(\cdot, x_0, \bar{\delta}, 0, T)$.

We define a *switching manifold* to be a compact set

$$M_{k,k-1} = \{x \,:\, \Phi_{k,k-1}(x) = 0\}\,, \tag{49}$$

in the state space of co-dimension-one, which is implicitly defined through an event function $\Phi_{k,k-1}$, for $k = 1, 2$. A trajectory starting at x_0 triggers the mode transition $0 \longrightarrow 1$ at the time $\tau(M_{10}, x_0)$, at which it first intersects M_{10}, i.e. $\Phi_{10}(x(\tau(M_{10}, x_0))) = 0$ and $\Phi_{10}(x(t)) \neq 0$, $\forall t \in [t_0, \tau)$. All subsequent intersections with M_{10} do not effect the system. Similar to the above, the union of all terminal points x_2 forms the terminal manifold M_{21}.

Assume that we use the switching manifolds as an explicit state-feedback controller, then under closed-loop operation, the switching times, the switch points and the activation duration are all functions of the x_0 and the switching manifolds (note that M_{10} only affects τ and δ_0, and M_{21} only affects T and δ_1) . We call switching manifolds M_{10} and M_{21} optimal, if the resultant activation duration $\bar{\delta}(M_{10}, M_{21})$ is a local optimal solution for a standard optimal control problem defined as

$$\min_{\bar{\delta}} J(\bar{\delta}, x_0) = \int_0^{\tau(M_{10})} L_0(x(t))dt + \int_{\tau(M_{10})}^{T(M_{21})} L_0(x(t))dt + \phi_1(x_1(M_{10}) + \phi_2(x_2(M_{21})), \tag{50}$$

for the entire set of initial conditions $x_0 \in \mathcal{X}_0$. Hence, optimal switching manifolds $M_{10}^\star$ and $M_{21}^\star$ characterize a family of optimal trajectories $x^\star(\cdot, x_0, \bar{\delta}^\star, 0, T^\star), x_0 \in \mathcal{X}_0$, where $\star$ denotes optimal entities with respect to the optimal control Problem (50).

The implicit representation (49) of the optimal surface by means of a switching condition is clearly not unique. One particular insightful representation of $\mathcal{M}_{k,k-1}^\star$ employs the event function

$$\delta_k^\star(x) = \Phi_{k+1,k}^\star(x)\,, \tag{51}$$

which returns the optimal activation duration $\delta_k^\star(x)$ for any current state x. If $\Phi_{k+1,k}^\star$ exists, it constitutes a *time-invariant* optimal feedback law for the optimal activation durations, which leads to the following lemma.

Lemma 4.1 *Stationary optimal manifolds $M^\star_{k+1,k}$ defined in (51), which solves the optimal control Problem (50), exist, if and only if there is a* time-invariant *state-feedback law for the optimal activation duration $\bar{\delta}^\star$.*

Proof 4.1 *First consider $k = 0$. Suppose that the optimal state-feedback law (51) for the first activation duration $\delta^\star_0$ is time-varying. Then there exists a time $t'_0 \neq t_0$ and a state x_0, so that the event function $\Phi^\star_{10}(x, t'_0) \neq \Phi^\star_{10}(x, t_0) > 0$ returns two different optimal activation durations for the first mode. Due to autonomy and time-invariance of the mode dynamics (48), both optimal trajectories $x^\star(t, x_0, \tau, 0, T)$, $x'^\star(t, x_0, \tau', 0, T)$ must switch at two different points $x_1 \neq x'_1$, which contradicts stationarity of the optimal surface $M^\star_{10}$.*

The other direction of the proof follows from inverting the arguments of the previous paragraph. The case for $k = 1$ is exactly the same.

Clearly, not all optimal control problems admit time-invariant optimal feedback laws. We now show that, time-invariant controller exists for a free final-time problem.

Theorem 4.1 *The optimal control Problem (50) admits stationary optimal switching manifolds $M^\star_{10}$ and $M^\star_{21}$, if the final time T is unspecified and the cost and constraint components L_i and ϕ_i are smooth, time-invariant functions.*

Proof 4.2 *First consider $M^\star_{10}$. By Bellman's Optimality Principle, an optimal trajectory $x^\star(t, x_0, \bar{\delta}^\star, 0, T)$ can be arbitrarily subdivided into two parts for some δ', where the trailing section $x^\star(t, x^\star(\delta'), (\delta^\star_0 - \delta'), [\delta^\star_0 - \delta', \delta^\star_1]^T, \delta', T)$ minimizes the costs-to-go $J^\star([\delta^\star_0 - \delta', \delta^\star_1]^T, x^\star(\delta'))$. Given $\delta' \leq \delta^\star_0$ the performance index*

$$J^\star(\bar{\delta}^\star, x_0) = \int_0^{\delta'} L_0(x(t))\mathrm{d}t + \int_{\delta'}^{\delta^\star_0} L_0(x(t))\mathrm{d}t + \int_{\delta^\star_0}^{\delta^\star_0+\delta^\star_1} L_1(x(t))\mathrm{d}t + \phi_1(x^\star_1) + \phi_2(x^\star_2) \tag{52}$$

can be split up into two parts. If L_i and ϕ_i are time-invariant, it is possible to shift the integration bounds of the cost-to-go term by $-\delta'$, which yields

$$J^\star(\bar{\delta}'^\star, x'_0) = \int_0^{\delta^\star_0-\delta'} L_0(x(t+\delta'))\mathrm{d}t + \int_{\delta^\star_0-\delta'}^{\delta^\star_0-\delta'+\delta^\star_1} L_1(x(t+\delta'))\mathrm{d}t + \phi_1(x^\star_1) + \phi_2(x^\star_2). \tag{53}$$

This expression obviously corresponds to the optimal costs incurred by the trajectory $x^\star(t, x_0', \tau^\star - \delta', 0, T - \delta')$ that starts at x_0'. As it took exactly δ' time units to get from x_0 to x_0', the system switches its mode at the same switch point $x_1^\star$ and $x_2^\star$ for both trajectories $x^\star(t, x_0, \tau^\star, 0, T)$ and $x^\star(t, x_0', \tau^\star - \delta', 0, T - \delta')$. Moreover, δ' is arbitrary, so that the optimal feedback law for $\delta_0^\star$ must be time-invariant and Lemma 4.1 ensures stationarity of $M_{10}^\star$. The case for $k = 1$ is exactly the same.

Clearly, for an optimal control problem with fixed final time, the cost functional defined in (50) is generally not constant when the activation durations are time-shifted. Thus, fixed final-time problems generally do not admit stationary optimal manifolds. Rather, the optimal switching manifolds and the switching planes are time-varying. Thus, henceforth in this chapter, we only consider fixed final-time problems.

In reference [61], we provided an algorithm for obtaining polytonal approximations of the often complex surfaces by off-line computations for a small set $\mathcal{X}_0$. However, the switching surfaces around a nominal trajectory for switched system with even small number of states require very large computational resources and is not feasible on-line. Hence, for a real-time controller, the switching surfaces are not used. Instead, we use the tangential approximations of the optimal switching surfaces, assuming that they are smooth, which they may not be in some cases. The switching planes are characterized by their normal direction vectors, which are also normal to the surfaces, and an offset vector. These planes are stationary since the switching surfaces are stationary. This approach therefore applies to systems with a much higher number of states and can be operated in a much large region of the state space.

4.3 State-feedback Control with Optimal Switching Planes

We now present our real-time robust state-feedback control algorithm for a free final-time optimal control problem. As mentioned before, we aim to use both explicit and implicit feedback control approaches. As a result, the control approach we propose is of a cascaded structure. We first propose the cascaded control approach with a switching plane controller, then we describe how we obtain the switching planes (in terms of the normal and the offset vectors) that are optimal with respect to an optimal control problem.

Note that since the problems we consider in this chapter are in the class of free final-time problems, while the rest of thesis mostly address fixed final-time problems, we sometimes have to introduce new notations and change the definitions of some notations used in previous chapters. They will be provided along the way.

4.3.1 Cascaded Control Approach

Consider the autonomous switched system

$$\dot{x}(t) \ = f_{q(t)}(x(t)) + w(t), \tag{54}$$

with initial condition $x(0) = x_0$, subject to unknown disturbances $w(t)$, where $q(t)$ denotes the operation mode of the switched system at time t, which is assumed to follow a predetermined finite sequence $\bar{q} = \{0, 1, \ldots, N-1\}$. Our switching law involves two basic components that operate on different time scales. The first component is a continuous-time *mode selector*, which implements a parametrized event function

$$\Phi(x(t), n(t), d(t)) = d(t) - n^T(t)x(t), \tag{55}$$

that can be evaluated with negligible computational effort. The mode selector generates a piecewise constant mode signal $q(t)$ by triggering a mode switch when $\Phi(x(t), n(t), d(t)) \leq 0$ and activating the next mode in the sequence $\bar{q}$. Suitable parameter trajectories $\{d(t)\}$ and $\{n(t)\}$ are provided by the second component, the *switching plane controller*

$$\begin{pmatrix} n^T(t) \\ d(t) \end{pmatrix} = h_c\left(x(t), q(t)\right). \tag{56}$$

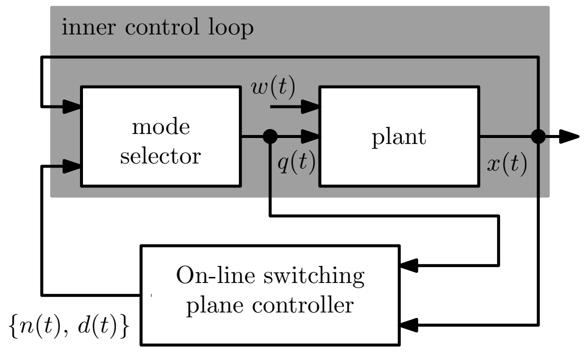


Figure 15: Cascaded control loop structure.

The switching plane controller consists of two discrete processes $\{n(t)\}$ and $\{d(t)\}$, which are updated every Δt time units, similar to $\{\bar{\tau}(t)\}$ in Chapter 3. At each sampling time t, this component computes adequate parameter values $n(t)$ (normal vector) and $d(t)$ (offset vector) based on the hybrid state $(x(t), q(t))$. Hence, the proposed control loop employs a real-time switching plane controller that is updated on-line.

Now we can propose a cascaded control loop, as shown in Figure 15. The *inner loop* operates in continuous time and consists of the plant and the mode selector. The inner loop and the switching plane controller form the *outer loop*, whose central task it is to ensures that the overall system meets user-defined specifications. The working principle of the real time switching plane controller and the overall control loop is illustrated in Figure 16 and can be described as follows: After the activation of the k-th mode, the state $x(t)$ evolves uncontrolled (and with noise $w(\cdot)$) according to the k-th modal function f_k and the disturbance $w(t)$. This evolution continues until satisfaction of the switching condition $\Phi(x(t), n(t), d(t)) \leq 0$ is detected. The exact switching time $\tau_{k+1} = \min_{t \geq \tau_k} t \ : \ d(t) - n^T(t)\, x(t) = 0$ indicates the first intersection of the state trajectory $x(\cdot)$ and the time-varying switching plane

$$\mathcal{T}(t) = \{x : d(t) - n^T(t)x = 0\} \tag{57}$$

in the state space, upon which a transition from $q(t)$ to $q(t) + 1$ is initiated, except if $q(t) = N - 1$. Subsequently, the same procedure is repeated for all remaining modes. If $q(t) = N - 1$, then the operation of the switched system is terminated. Provided a sufficiently high update frequency and under reasonable assumptions, the switching plane controller incrementally shifts and rotates $\mathcal{T}(t)$ in state space during operation of the plant. By the nature of the real-time problem we propose, we often only consider the trajectory starting at time t and ending at time τ_N.

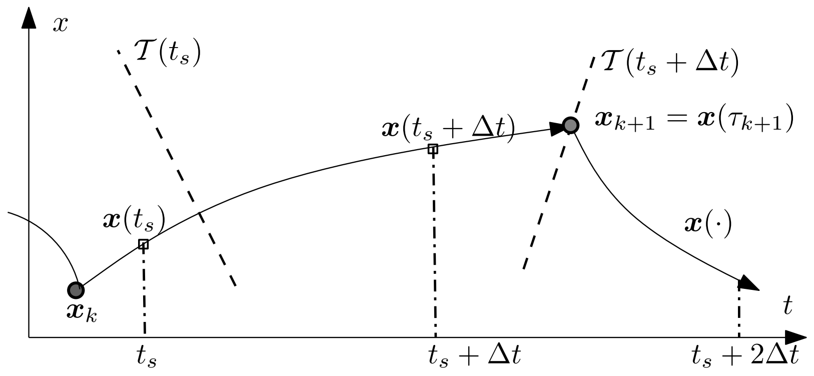


Figure 16: Working principle of the real-time switching plane controller. At a sample time t, the switching plane $\mathcal{T}(t)$, parameterized by the normal $n(t)$ and the offset vector $d(t)$, is updated. Similarly at the time $t + \Delta t$. A mode transition occurs whenever the state trajectory intersects with the time-varying switching plane $\mathcal{T}(t)$. In this case, this intersection occurred during the time interval $[t + \Delta t, t + 2\Delta t]$.

For compactness, let $\bar{\tau} = \{\tau_k\}_{k=1}^{N}$ and $\bar{x} = \{x_k\}_{k=1}^{N}$ denote the *switching time sequence* and the *switching point sequence*, which contains all switching points $x_k = x(\tau_k)$. Moreover, we call $\delta_k = \tau_{k+1} - \tau_k$ the activation duration of mode q_k and $\bar{\delta} = \{\delta_k\}_{k=0}^{N-1}$ the corresponding *activation duration sequence*.

4.3.2 Real-time Optimal Control Problem

The cascaded control loop proposed can use any switching planes, but we want to use switching planes that are optimal with respect to the optimal control problem at hand. One may first formulate a standard optimal control problem. Given an autonomous switched system (54) and a finite mode sequence $\bar{q}$, we aim to determine a switching plane controller (56), such that the closed-loop generates a switching time vector $\bar{\tau} = [\tau_1, \tau_2, ..., \tau_N]^T$, which solves the following problem

$$\min_{\bar{\tau}} J_{\bar{q}}(x_0, \bar{\tau}) = \sum_{k=0}^{N-1} \left(\int_{\tau_k}^{\tau_{k+1}} L_k(x(t)) \mathrm{d}t + \phi_k(x_k) \right) + \phi_N(x_N) \tag{58}$$

subject to the dynamics (54) without noise ($w(t) = 0, \forall t$) and

$$0 \leq \tau_1, ..., \leq \tau_N, \tag{59}$$

for any initial state $x_0 \in \mathcal{X}_0$. Functions L_k encode instantaneous costs, which assess the transient behavior, ϕ_k accounts for potential switching cost and ϕ_N represents terminal costs. Trajectory cost functions L_k are required but switching cost functions ϕ_k can be zero. Since we only consider fixed final-time problems, we denote the final time T as τ_N throughout this chapter.

A trajectory $x^\star(\cdot)$, associated to a local minimizer $\bar{\tau}^\star$ of (58), is denoted as the associated *nominal* trajectory. The proposed problem without disturbances $w(t)$ is a standard real-time optimal control problem solved in Chapter 3. Our optimal-control based maps and (56), however, are explicitly designed to provide a robust loop behavior under disturbances, and go beyond solving Problem (58) without noise. Our controller provides correction responses for systems under disturbances, and guides the system back to an optimal trajectory with respect to Problem (58). Hence, we consider to solve this problem over a set of initial conditions, which corresponds to a family of trajectories in the neighborhood of a nominal trajectory. Hence, we shall reformulate the problem into a real-time problem, where the state trajectory is assumed to be effected by the noise $w(\cdot)$, and the control task at time t is to regulate the remaining state trajectory for the remaining modes of operation.

For the subsequent presentation of the controller design procedure, we need to explicitly distinguish between *elapsed* trajectories, i.e. signals that have been implemented by the system in the past, and *predicted* trajectories over a finite interval into the future. We thus define a set of new notations.

Let us first introduce a *relative time frame* r, which slides along the absolute time axis and indicates the *time difference* to current time t. Given the hybrid state $(x(t),\, q(t) = j)$, with j being the index of the currently activated mode, and a sequence of fixed *predicted switching times* $\bar{\rho}(t) = [\rho_1(t), \rho_2(t), ..., \rho_{N(t)}(t)]^T$ for the remaining *predicted mode sequence* $\bar{\theta}(t) = \{j, j+1, ..., N-1\}$ of length $N(t) = N - j$ to be implemented in the future, we can generate a corresponding *predicted mode signal* $\theta(r|t)$ and use the disturbance-free state space model

$$\dot{\xi}(r|t) \ = f_{\theta(r|t)}(\xi(r|t))\,, \ \ \xi(0|t) = x(t) \tag{60}$$

to simulate the *predicted state trajectory* $\xi(r|t)$ over a finite duration into the future. Denote $\rho_0(t) = t$. Define $\bar{\chi}(t)$ such that $\chi_k(t) = \rho_{k+1}(t) - \rho_k(t), k = 0, 1, ..., N(t) - 1$. For consistency, we call $\bar{\chi}(t)$ the *predicted activation duration sequence*, which is dual to $\bar{\rho}(t)$ and refer to $\bar{\xi}(t)$ as the *predicted switching point sequence.* Additionally, let us introduce the shifted sequence $\bar{\chi}(\Delta t|t)$, where the first activation duration $\bar{\chi}_0(\Delta t|t) = \chi_0(t) - \Delta t$ is shifted by Δt time units, whereas the remaining duration $\bar{\chi}_k(\Delta t|t) = \chi_k(t)$ for all $k = 1 \ldots (N(t) - 1)$. An example of an elapsed trajectory $x(\cdot)$ and the attached prediction $\xi(\cdot|t)$ for a sample time $t = t_s$ is shown in Figure 17.

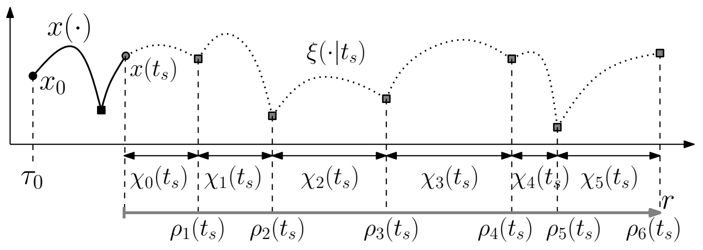


Figure 17: Example of an elapsed state trajectory $x(\cdot)$ and its prediction $\xi(\cdot|t)$ (dotted line) at time $t = t_s$.

Thus, all predicted signals and variables are explicitly associated to the time t and the corresponding hybrid state $(x(t), q(t))$. In contrast to their elapsed counterparts τ_k (δ_k), all $\rho_k(t)$ (all $\chi_k(t)$) are considered to be independent of the past state and disturbance history and only dependent on the current state $x(t)$. Furthermore, we define the feasible set $\Lambda_\rho(t) = \{\bar{\rho}(t) : t = \rho_0(t) \leq \rho_1(t) \leq \ldots, \leq \rho_{N(t)-1}(t) \leq \rho_{N(t)}(t)\}$ and equivalently, $\Lambda_\chi(t) = \{\bar{\lambda}(t) : \lambda_k(t) \geq 0, k = 0, 1, ..., N(t) - 1\}$. Finally, we denote $n(\cdot)$ and $d(\cdot)$ as trajectories of the switching plane parameters. Thus, $n_k(t) = n(\rho_{k+1}(t))$ and $d_k = d(\rho_{k+1}(t))$ constitute sequences $\bar{n}(t)$ and $\bar{d}(t)$. Note that, If we have the trajectories $n(\cdot)$ and $d(\cdot)$, then we can freely integrate the simulated state trajectory forward and obtain the activation duration $\bar{\chi}(t, \bar{n}(t), \bar{d}(t))$.

With these definitions, we are in the position to formulate our desired real-time optimal control problem, which is

$$\min_{\bar{\rho}(t) \in \Lambda_\rho(t)} J_{\bar{\theta}(t)}(x(t), \bar{\rho}(t)) = \sum_{j=0}^{N(t)-1} \left(\int_{\rho_k(t)}^{\rho_{k+1}(t)} L_k(\xi(r|t)) \mathrm{d}r + \phi_k(\xi_k(t)) \right) + \phi_N(\xi_{N(t)}(t)), \quad (61)$$

subject to the dynamics (60),

where L_k, ϕ_k, ϕ_N are all identical to the functions of the original Problem (58). The foundation for the determination of suitable values $n(t)$ and $d(t)$ at time t is the knowledge of the triple $\big(x(t), \bar{\rho}(t), \bar{\theta}(t)\big)$. In particular, since the optimal switching planes are anchored at the optimal switching points, we see that the optimal values $n^\star(t)$ and $d^\star(t)$ are obtained, if and only if $\bar{\rho}(t)$ constitutes a local minimizer for Problem (61).To ensure well-posedness of the problem and to simplify notations, we make the following assumptions throughout the remaining sections of this chapter:

4.1 We assume that the state information is obtained without measurement error. Measurement noise can be treated as process disturbances.

4.2 The modal functions f_k are all three-times continuously differentiable.

4.3 L_{q_k}, ϕ_{q_k}, ϕ_N are three-times continuously differentiable.

4.3.3 Exact Discretization of the Optimal Control Problem

Now we are ready to design a real-time algorithmic framework and summarize our procedure for incrementally adapting the parametrized switching plane $\mathcal{T}(t)$. The key for obtaining the optimal switching plane orientation $n(t)$ and offset $d(t)$, is to apply a novel approach, transforming the optimization problem into an equivalent well-known discrete-time optimal control problem via an exact discretization process. This process reformulate Problem (61) into the following discrete-time optimal control problem

$$\min_{\bar{\chi}(t)\in\Lambda_{\chi}(t)} \tilde{J}_{\bar{\theta}(t)}(x(t),\bar{\chi}(t)) = \sum_{k=0}^{N(t)} g_k(\xi_k(t),\chi_k(t)) \quad \text{subject to} \tag{62}$$

$$\xi_{k+1}(t) = h_k(\xi_k(t),\chi_k(t)), \quad \xi_0(t) = x(t), \tag{63}$$

in terms of the predicted activation duration sequence $\bar{\chi}(t)$ with discretized dynamics and costs

$$h_k(\xi_k(t),\chi_k(t)) = \xi_k + \int_0^{\chi_k(t)} f_{\theta_k(t)}(\xi(r|t))\mathrm{d}r \tag{64}$$

$$g_k(\xi_k(t),\chi_k(t)) = \int_0^{\chi_k(t)} L_{\theta_k(t)}(\xi(r|t))\mathrm{d}r + \phi_{\theta_k(t)}(\xi_k(t)) \tag{65}$$

$$g_{N(t)}(\xi_k(t),\chi_k(t)) = \phi_N(\xi_{N(t)}(t)). \tag{66}$$

Note that closed-form expressions of (64), (65) can generally not be derived, but both functions can always be evaluated via simulation. In contrast to its continuous-time counterpart, the transformed Problem (62), which we will simply refer to as $\Psi(x(t),\bar{\theta}(t),t)$, constitutes a standard nonlinear optimization problem. Hence, it can be efficiently solved by one of numerous existing iterative procedures such as *Sequential Quadratic Programming* (SQP) [28]. Most importantly, the concept of neighboring extremal solutions by a *perturbation feedback controller* is well-defined for such a problem setting [42].

However, it should be noted that, although the problem is transformed to a nonlinear optimal control problem, the results involving nonlinear optimal control with respect to the control input have to be re-interpreted in terms of a switched system, since the control input $\bar{\chi}(t)$ in this case are time durations, not control input in the standard sense. The actual control action is not applied until the time duration becomes 0. Hence, we must interpret the results in light of this important structure of the switched system. Hence, one central novelty of this chapter is that we apply perturbation feedback control from nonlinear optimal control to a switched system timing optimal control problem, and interpret the resultant controller as the optimal switching plane controller.

To exploit both above mentioned aspects in our controller design procedure, we need closed-form expressions of the first and second order partial derivatives of (64) and (65). Therefore, let us introduce the Hamiltonian

$$H_k(\xi_k(t), \chi_k(t), p_{k+1}(t)) = p_{k+1}^{\mathrm{T}}(t) h_k(\xi_k(t), \chi_k(t)) + g_k(\xi_k(t), \chi_k(t)), \tag{67}$$

and the costate $p_k(t)$ at time t determined via backward iteration

$$p_k^{\mathrm{T}}(t) = p_{k+1}^{\mathrm{T}}(t) A_k(t) + a_k(t), \;\; p_N(t) = 0. \tag{68}$$

A_j and a_j are obtained from forward integration required for the predicted state trajectory $\xi(r|t)$.

$$A_k(t) = \frac{\partial h_k}{\partial \xi}(\xi_k(t), \chi_k(t)) = Z_k(\chi_k(t)), \tag{69}$$

$$a_k(t) = \frac{\partial g_k}{\partial \xi}(\xi_k(t), \chi_k(t)) = \int_0^{\chi_k(t)} \frac{\partial L_k}{\partial \xi}(\zeta(r|t)) Z_k(r|t) \mathrm{d}r + \frac{\partial \phi_k}{\partial x}(\xi_k(t)) \tag{70}$$

where the state transition matrix $Z_k(r|t)$ and $\zeta(r|t)$ follows from simultaneous integration of

$$\dot{\zeta}(r|t) = f_k(\zeta(r|t)), \zeta(0) = \xi_k(t) \tag{71}$$

$$\dot{Z}_k(r|t) = \frac{\partial f_k}{\partial \zeta}(\zeta(r|t)) Z_k(r|t), \;\; Z(0) = I. \tag{72}$$

Moreover, let us introduce the abbreviations

$$B_k(t)=\frac{\partial h_k}{\partial \chi}(\xi_k(t),\chi_k(t)),\quad b_k(t)=\frac{\partial g_k}{\partial \chi}(\xi_k(t),\chi_k(t)) \tag{73}$$

$$Q_k(t)=\frac{\partial^2 H_k}{\partial \xi^2}(\xi_k(t),\chi_k(t),p_{k+1}(t)) \tag{74}$$

$$M_k(t)=\frac{\partial H_k^2}{\partial \xi \partial \chi}(\xi_k(t),\chi_k(t),p_{k+1}(t)) \tag{75}$$

$$R_k(t)=\frac{\partial H_k^2}{\partial \chi^2}(\xi_k(t),\chi_k(t),p_{k+1}(t))=M_k(t)A_k^{-1}(t)B_k(t), \tag{76}$$

$$k=0,1,...,N(t)-1$$

where the relation (76) between M_k and R_k plays a crucial role later on. It results from the structure of the optimization problem and can be verified by differentiations.

Finally, let us also introduce the augmented performance index

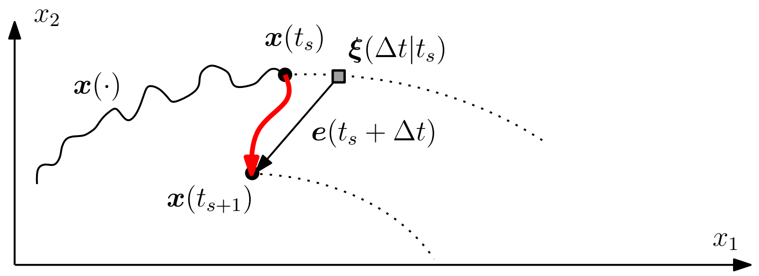
$$\begin{aligned}\tilde{J}'_{\bar{\theta}(t)}(x(t),\bar{\chi}(t)) &= \phi_{N(t)}(\xi_{N(t)}(t)) - p^T_{N(t)}(t)\xi_{N(t)} + H_0(\xi_0(t),\chi_0(t),p_1(t)) \\ &\quad + \sum_{k=1}^{N(t)-1}(H_k(\xi_k(t),\chi_k(t),p_{k+1}(t)) - p_k^T(t)\xi_k(t)), \end{aligned} \tag{77}$$

which is obtained by adjoining the constraints (63) to (62). As known in nonlinear optimal control, this augmentation transforms the constrained optimization problem into an equivalent unconstrained one.

4.3.4 Robust State-feedback Optimal Control Algorithm

Now we are ready to present the method to apply a neighboring extremal control scheme for the discretized Problem (62), and see that they can be constructed by the optimal switching planes. Furthermore, we are able to obtain the normal $n(t)$ and offset vector $d(t)$.

The previous exact discretization process comprises two central benefits: First, due to [28, 36] Sequential Quadratic Programming (SQP) provides the means for obtaining the Newton direction $\Delta\bar{\chi}(t)$ (which is equivalent to the Newton direction for $\Delta\bar{\rho}(t)$), namely by solving a constrained discrete-time LQR problem. Second,



x_2
$\boldsymbol{x}(t_s)$
$\boldsymbol{\xi}(\Delta t|t_s)$
$\boldsymbol{x}(\cdot)$
$\boldsymbol{e}(t_s+\Delta t)$
$\boldsymbol{x}(t_{s+1})$
x_1

Figure 18: Prediction error $e(t)$ at a sample time time $t = t_s$ in a noisy environment. It represents the noise accumulated during the time interval $[t_s, t_s + \Delta t]$.

this standard optimal control problem yields a well-defined notion of a neighboring extremal solution (see [42]), which can be exploited for the determination of a suitable plane normal $n(t)$. To see both aspects, denote the prediction error (due to noise) at time $t + \Delta t$ by $e(t + \Delta t) = x(t + \Delta t) - \xi(\Delta t|t)$. This term represents the noise accumulated during the time interval $[t, t + \Delta t]$ (refer to Figure 18).

A second order Taylor expansion of the augmented cost functional (77) around the current prediction $\bar{\chi}(\Delta t|t)$ for the next sampling time $t + \Delta t$ and the corresponding state prediction $\xi(\Delta t|t)$ yields

$$\begin{aligned}
&\tilde{J}'_{\bar{\theta}(t)}(\xi(\Delta t|t) + e(t+\Delta t), \bar{\chi}(\Delta t|t) + \Delta\bar{\chi}'(t+\Delta t)) \\
= \ &\tilde{J}'_{\bar{\theta}(t)}(\xi(\Delta t|t), \bar{\chi}(\Delta t|t)) + \frac{\partial \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}}(\xi(\Delta t|t), \bar{\chi}(\Delta t|t))\Delta\bar{\chi}'(t+\Delta t) \\
&+\frac{\partial \tilde{J}'_{\bar{\theta}(t)}}{\partial \xi}(\xi(\Delta t|t), \bar{\chi}(\Delta t|t))e(t+\Delta t) + \frac{1}{2}[e^{\mathrm{T}}(t+\Delta t) \quad \Delta\bar{\chi}'(t+\Delta t)^{\mathrm{T}}] \\
&\begin{bmatrix} \frac{\partial^2 \tilde{J}'_{\bar{\theta}(t)}}{\partial \xi^2}(\xi(\Delta t|t), \chi(\Delta t|t)) & \frac{\partial^2 \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}\partial \xi}(\xi(\Delta t|t), \chi(\Delta t|t)) \\ \left(\frac{\partial^2 \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}\partial \xi}(\xi(\Delta t|t), \chi(\Delta t|t))\right)^T & \frac{\partial^2 \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}^2}(\xi(\Delta t|t), \chi(\Delta t|t)) \end{bmatrix} \begin{bmatrix} e(t+\Delta t) \\ \Delta\bar{\chi}'(t+\Delta t) \end{bmatrix} \\
&+O\left(\left\| \begin{bmatrix} e(t+\Delta t) \\ \Delta\bar{\chi}'(t+\Delta t) \end{bmatrix} \right\|^2\right). \qquad (78)
\end{aligned}$$

The subsequent Newton direction $\Delta\bar{\chi}(t+\Delta t)$ associated with the pair $(\xi(\Delta t|t), \bar{\chi}(\Delta t|t))$, which explicitly accounts for the disturbance effects at the next instant $t + \Delta t$, is the solution to

$$\min_{\Delta\bar{\chi}'(t+\Delta t)\in\Lambda_\chi(t+\Delta t)} \tilde{J}'_{\bar{\theta}(t)}(\xi(\Delta t|t)+e(t+\Delta t),\bar{\chi}(\Delta t|t)+\Delta\bar{\chi}'(t+\Delta t)). \tag{79}$$

This is the central idea of SQP, namely reducing the nonlinear optimal control problem into a LQR problem (see [28])

$$\begin{aligned}\min_{\Delta\bar{\chi}\in\Lambda_\chi(t)}\Bigg(\sum_{k=0}^{N(t)-1}&\Big(\frac{1}{2}\Delta\xi_k^{\mathrm{T}}Q_k(t)\Delta\xi_k+a_k^{\mathrm{T}}(t)\Delta\xi_k\\&+\Delta\chi_k^{\mathrm{T}}M_k(t)\Delta\xi_k+\frac{1}{2}\Delta\chi_k^{\mathrm{T}}R_k(t)\Delta\chi_k+b_k^{\mathrm{T}}(t)\Delta\chi_k\Big)\\&+(\frac{1}{2}\xi_{N(t+\Delta t)}^{\mathrm{T}}Q_{N(t)}(t)+a_{N(t)}^{\mathrm{T}}(t))\Delta\xi_{N(t+\Delta t)}\Bigg)\end{aligned} \tag{80}$$

subject to

$$\Delta\xi_{k+1}=A_k(t)\Delta\xi_k+B_k(t)\Delta\chi_k,\quad \Delta\xi_0=e(t+\Delta t), \tag{81}$$

where $A_k(t)$, $B_k(t)$, $a_k(t)$, $b_k(t)$, $Q_k(t)$, $M_k(t)$, $R_k(t)$ are all defined in the last section as partial derivatives of the Hamiltonian (67). Henceforth we refer to this problem as LQR $(x(t+\Delta t),\bar{\chi}(t))$. Using the perturbation feedback controller presented in [36, 42], we can use the LQR state-feedback gain $k_k^T(t)$ obtained from solving LQR $(x(t+\Delta t),\bar{\chi}(t))$ to determine a sequence of the normal vector $n_k(t)$ and a sequence of the offset $d_k(t)$

$$n_k^T(t)=k_k^T(t)A_k^{-1}(t),\quad d_k(t)=n_k^T(t)\xi_{k+1}(t). \tag{82}$$

The effect of state intersection with the switching plane with the normal and offset vectors defined above can be seen as a perturbation feedback control

$$\bar{\chi}(t,\bar{n}(t),\bar{d}(t))=\bar{\chi}(t)+H^{-1}(t)\frac{\partial^2\tilde{J}'_{\bar{\theta}(t)}}{\partial\xi\partial\bar{\chi}}(\xi(t),\bar{\chi}(t))e(t+\Delta t), \tag{83}$$

due to the following fact from [42]

$$u_k^TH^{-1}(t)\frac{\partial^2\tilde{J}'_{\bar{\theta}(t)}}{\partial\xi\partial\bar{\chi}}(\xi(t),\bar{\chi}(t))e(t+\Delta t)=k_k^T(t)\Delta\xi_k, \tag{84}$$

where u_k denotes the k-th unit vector. This allows for evaluating the perturbation control law (83) at time $t + \Delta t$ simply by means of an event-driven forward-in-time simulation of the plant dynamics with initial state $x(t + \Delta t)$. Thus, compared to the feedback approach in Chapter 3, the determination of switching plane parameters and the evaluation of the perturbation controller (83) only causes marginal additional effort, whereas the potential for improving the loop performance is significant. The set of planes defined by $\bar{n}(t)$ and $\bar{d}(t)$ perform a Newton step correction on the error $e(t+\Delta t)$. This correction is done for every iteration as long as one uses the switching plane controller, which mounts to better performance.

When the activation duration vector $\bar{\chi}^{\star}(t)$ is optimal, the vector $n_k^{\star}(t)$ is normal to the optimal switching surface. This is resulted by re-interpreting the perturbation feedback control corrections in the setting of the activation durations. After the sequences $\bar{n}(t)$ and $\bar{d}(t)$ are obtained, we always use the first upcoming plane parameters for $n(t)$ and $d(t)$. Thus $n(t) = n_0(t)$ and $d(t) = d_0(t)$.

With the above results for the normal $\bar{n}(t)$ and offset vector $\bar{d}(t)$, we can propose the overall real-time algorithm.

Algorithm 4.1 *Step 0: Set* $t = 0$, $\bar{\theta}(0) = \bar{q}$, *specify tolerance* $\varepsilon > 0$ *and step size* Δt. *Determine a feasible initial sequence* $\bar{\chi}(0)$ *for given* x_0.

Iterate until mode N *gets deactivated at* $t > \tau_N = T$

Step 1: At current time t*, obtain new measurement* $x(t)$.

Step 2: If $\chi_0(t) > \Delta t$*, then*

Step 3.1: Simulate and obtain $\xi(r|t)$ *by forward integration and the switching point sequence* $\xi_k(t)$*, using the switching planes parameters* $\bar{n}(t)$ *and* $\bar{d}(t)$*. During this event-triggered forward integration, the sequence* $\bar{\chi}(t - \Delta t) = \bar{\chi}(t, \bar{n}(t), \bar{d}(t))$ *is obtained, which includes the corrective effects of the perturbation feedback controller* (83).

Step 3.2 Compute matrix sequences $Q_k(t)$, $M_k(t)$, $R_k(t)$, $A_k(t)$, $k = 0, 1, ..., N(t)-1$. *Solve* $LQR(x(t), \bar{\chi}(t - \Delta t))$ *to obtain the intermediate update* $\Delta\bar{\chi}(t + \Delta t)$*, update*

$\bar{n}(t)$, $\bar{d}(t)$ from (82)*. Substitute $n(t) = n_0(t)$ and $d(t) = d_0(t)$ into the event function* (55)*.*

Step 3.3: While executing steps 3.1-2, check for collision of $x(\cdot)$ with $\mathcal{T}(t)$, i.e. for $d(t) - n^T(t)x(t) < -\varepsilon$. If so, interrupt all computations, set $\Delta\chi(t+\Delta t) = 0$ and jump to (4.1).

Step 3.4: Wait until Δt time units have elapsed and return.

else

Step 4.1: Wait until the collision of $x(\cdot)$ with $\mathcal{T}(t)$ is detected at time $\tau_{q(t)+1} \geq t$. Set $t + \Delta t = \tau_{q(t)+1}$ and $\Delta\chi(t+\Delta t) = \Delta\chi(t)$.

Step 4.2: Set $N(t+\Delta t) = N(t) - 1$, shrink mode sequence $\bar{\theta}(t+\Delta t) = \{\theta_k(t)\}_{k=1}^{N(t+\Delta t)}$, and trigger mode transition $q(t) \to q(t) + 1$.

The main difference between this algorithm and Algorithm 3.1 proposed in Chapter 3 is that, for Algorithm 4.1, there are two Newton corrections (one on the state deviation, one on the activation duration deviation) done for each step of this algorithm. The extra Newton correction on the state deviation is achieved by the neighboring extremal solutions obtained with the implementation of the optimal switching plane controller. This algorithm thus provides a more robust controller than Algorithm 3.1 in terms of minimizing the cost functional (61). Note that the computation load of each step of the algorithm is similar to that of Algorithm 3.1, since one step of the Newton direction must be computed.

However, this controller is more suitable for small deviations from the optimal trajectory, since Newton methods are not globally convergent, although they admit very fast convergence rate (quadratic) when the state trajectory is close to the nominal optimal trajectory. Performance of Algorithm 4.1 depends on the frequency and amplitude of the disturbances, and it also depends on the number of iterations executed by this real-time algorithm. The performance analysis of this algorithm is addressed in the next sub-section.

4.3.5 Convergence Analysis

In this section, we present convergence analysis of the proposed algorithm. Similar to Chapter 3, we characterize the performance of the algorithm not in an asymptotic sense, but rather in the sense of convergence rate, due to the real-time nature of the algorithm.

We first note a stationarity property follows from Bellman's Principle of Optimality. If $\bar{\chi}(t)$ is locally optimal at a time t, then the update of step 1.4 conserves optimality for all future times if there is no noise. This property follows from stationarity of the optimal switching points in the absence of disturbances.

Lemma 4.2 *Let $\bar{\chi}^\star(t_0)$ be a local minimizer of $\Psi(x(t_0), \bar{q}(t_0), t_0)$, then $\bar{\chi}^\star(t)$ with $\chi_0^\star(t) = \chi_j^\star(t_0) - (t - \rho_j(t_0))$ and $\chi_k^\star(t) = \chi_{k+j}^\star(t_0)$, $k = 1 \dots N(t)$ constitutes a local minimizer of the cost-to-go problem $\Psi(\xi(t - t_0|t_0), \bar{\theta}(t), t)$ for any t_0 and any $t \in [\rho_j(t_0), \rho_{j+1}(t_0)]$ with $N(t) = N - j$ and $\bar{\theta}(t) = \{j, j+1, ...N-1\}$.*

Proof 4.3 *The result of Lemma 4.2 follows directly from Bellman's Principle of Optimality and the assumed time invariance of L_{q_k}, ϕ_{q_k} and ψ.*

Consequently, the overall undisturbed loop behaves as specified in Problem (58). Now let us investigate central properties of the real-time approach in the presence of disturbances. Suppose that at time t, a local minimizer $\bar{\chi}^\star(t)$ corresponding to $x(t)$ is known. If disturbances perturb the execution over $[t, t+\Delta t]$, then the predicted local minimizer $\bar{\chi}^\star(\Delta t|t)$ is no longer a minimizer with respect to $x(t+\Delta t)$ at time $t+\Delta t$. Nevertheless, the difference between $\bar{\chi}^\star(\Delta t|t)$ and the true minimizer $\bar{\chi}^\star(t+\Delta t)$ is bounded. Recall the prediction error (due to noise) at time t is obtained by $e(t+\Delta t) = x(t+\Delta t) - \xi(\Delta t|t)$ (see Figure 18), we can derive the following relation.

Theorem 4.2 *If no mode transition occurs between t and $t + \Delta t$ and the gradient $\frac{\partial \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}}(x, \bar{\chi})$ is differentiable at points $(\xi(\Delta t|t), \bar{\chi}^\star(\Delta t|t))$ and $(\xi(\Delta t|t), \bar{\chi}^\star(t+\Delta t))$,* *then there exists a bounded* $K > 0$*, such that*

$$||\bar{\chi}^\star(t+\Delta t) - \bar{\chi}^\star(\Delta t|t)|| \leq K||e(t+\Delta t)|| \ . \tag{85}$$

Proof 4.4 *Since no mode transition is triggered during time interval* $[t, t+\Delta t]$ *and because of Lemma 4.2, we know that* $\bar{\theta}(t) = \bar{\theta}(t+\Delta t)$ *and that the augmented performance index* (77) *satisfies*

$$\begin{aligned}
&\frac{\partial \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}}(x(t), \bar{\chi}^\star(t)) = 0, \ \ \frac{\partial \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}}(\xi^\star(\Delta t|t), \bar{\chi}^\star(\Delta t|t)) = 0 \\
&\frac{\partial \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}}(x(t+\Delta t), \bar{\chi}^\star(t+\Delta t)) = 0 \ .
\end{aligned} \tag{86}$$

A Taylor expansion for the latter term (86) *yields*

$$\begin{aligned}
& \frac{\partial \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}}(x(t+\Delta t), \bar{\chi}^\star(t+\Delta t)) \\
= \ & \frac{\partial \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}}(\xi^\star(\Delta t|t), \bar{\chi}^\star(t+\Delta t)) \\
& + \frac{\partial^2 \tilde{J}'_{\bar{\theta}(t)}}{\partial \xi \partial \bar{\chi}}(\xi^\star(\Delta t|t), \bar{\chi}^\star(t+\Delta t)) e(t+\Delta t) + O(||e(t+\Delta t)||^2) \\
= \ & \frac{\partial^2 \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}^2}(\xi^\star(\Delta t|t), \bar{\chi}^\star(\Delta t|t))(\bar{\chi}^\star(t+\Delta t) - \bar{\chi}^\star(\Delta t|t)) \\
& + \frac{\partial^3 \tilde{J}'_{\bar{\theta}(t)}}{\partial \xi \partial \bar{\chi}^2}(\ldots) \, e(t+\Delta t)(\bar{\chi}^\star(t+\Delta t) - \bar{\chi}^\star(\Delta t|t)) \\
& + \frac{\partial^2 \tilde{J}'_{\bar{\theta}(t)}}{\partial \xi \partial \bar{\chi}}(\xi^\star(\Delta t|t), \bar{\chi}^\star(\Delta t|t)) e(t+\Delta t) \\
& + O(||e(t+\Delta t)||^2) + O(||\bar{\chi}^\star(t+\Delta t) - \bar{\chi}^\star(\Delta t|t)||^2) = 0.
\end{aligned} \tag{87}$$

From positive definiteness of the Hessian

$$H^\star(t) = \frac{\partial^2 \tilde{J}_{\bar{\theta}(t)}}{\partial \bar{\chi}^2}(\xi^\star(\Delta t|t), \bar{\chi}^\star(\Delta t|t)) = \frac{\partial^2 \tilde{J}_{\bar{\theta}(t)}}{\partial \bar{\chi}^2}(\xi^\star(t), \bar{\chi}^\star(t)), \tag{88}$$

and the assumption that $\|e(t+\Delta t)\| \ll 1$ *we get invertibility of* $H^\star(t) + \frac{\partial^3 \tilde{J}_{\bar{\theta}(t)}}{\partial \xi \partial \bar{\chi}^2}(\ldots)e(t+$

$\Delta t)$. *Therefore,*

$$\begin{aligned}
&\bar{\chi}^\star(t+\Delta t) - \bar{\chi}^\star(\Delta t|t) \\
= \quad & -\left(H^\star(t) + \frac{\partial^3 \tilde{J}'_{\bar{\theta}(t)}}{\partial\xi\partial\bar{\chi}^2}(\ldots)e(t+\Delta t)\right)^{-1}\left(\frac{\partial^2 \tilde{J}'_{\bar{\theta}(t)}}{\partial\xi\partial\bar{\chi}}(\xi^\star(\Delta t|t), \bar{\chi}^\star(\Delta t|t))\, e(t+\Delta t)\right. \\
&\left. + O(||e(t+\Delta t)||^2) + O(||\bar{\chi}^\star(t+\Delta t) - \bar{\chi}^\star(\Delta t|t)||^2)\right), \qquad (89)
\end{aligned}$$

which after application of the perturbation lemma for inverse matrices ([16], Fact 9.9.43) yields

$$\begin{aligned}
\bar{\chi}^\star(t+\Delta t) - \bar{\chi}^\star(\Delta t|t) \quad = \quad & -(H^\star(t))^{-1}\frac{\partial^2 \tilde{J}'_{\bar{\theta}(t)}}{\partial\xi\partial\bar{\chi}}(\xi^\star(\Delta t|t), \bar{\chi}^\star(\Delta t|t))\, e(t+\Delta t) \\
& + O(||e(t+\Delta t)||^2) + O(||\bar{\chi}^\star(t+\Delta t) - \bar{\chi}^\star(\Delta t|t)||^2).
\end{aligned} \qquad (90)$$

This completes the proof.

Theorem 4.2 quantifies the degradation of $\bar{\chi}(t)$ in between two consecutive update steps, if we skip the Newton correction. Executing this correction and again assuming that the deviation from the predicted path $\|e(t)\| \ll 1$, we can examine the convergence rate of Algorithm 4.1 by the following theorem.

Theorem 4.3 *Suppose that no mode transition occurs between t and $t+\Delta t$ and that the gradient $\frac{\partial \tilde{J}'_{\bar{\theta}(t)}}{\partial\bar{\chi}}(x, \bar{\chi})$ is differentiable at points $(\xi(\Delta t|t),\ \bar{\chi}^\star(\Delta t|t))$ and $(\xi(\Delta t|t),\ \bar{\chi}^\star(t+\Delta t))$. If the Hessian $H(t) = \frac{\partial^2 \tilde{J}'_{\bar{\theta}(t)}}{\partial\bar{\chi}^2}(x(t), \bar{\chi}(t))$ is positive definite at time $t+\Delta t$, then there exists a bounded constant $K > 0$ such that*

$$||\bar{\chi}(t+\Delta t) - \bar{\chi}^\star(t+\Delta t)|| \leq K\left(\|\bar{\chi}(t) - \bar{\chi}^\star(t)\|^2 + \|e(t+\Delta t)\|\right) \qquad (91)$$

Proof 4.5 *Suppose that no mode transition is initiated in between $[t, t+\Delta t]$ so that $\bar{\theta}(t) = \bar{\theta}(t+\Delta t)$ and that the previously predicted optimal activation duration sequence is updated at time $t+\Delta t$ according to*

$$\bar{\chi}(t+\Delta t) = \bar{\chi}(\Delta t|t) - \left(\frac{\partial^2 \tilde{J}'_{\bar{\theta}(t)}}{\partial\bar{\chi}^2}(\xi(\Delta|t), \bar{\chi}(\Delta t|t))\right)^{-1}\frac{\partial \tilde{J}'_{\bar{\theta}(t)}}{\partial\bar{\chi}}(x(t+\Delta t), \bar{\chi}(\Delta t|t)). \qquad (92)$$

For convenience

$$H(t) = \frac{\partial^2 \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}^2}(\xi(\Delta t|t), \bar{\chi}(\Delta t|t), \tag{93}$$

and $\nabla \tilde{J}'(t) = \frac{\partial \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}}(x(t), \bar{\chi}(\Delta t|t-\Delta t))$ *denote the Hessian and the gradient evaluated for the predicted activation duration sequence* $\bar{\chi}(\Delta t|t-\Delta t)$ *of the previous update step and let* $\Delta\bar{\chi}(t) = \bar{\chi}(t) - \bar{\chi}^\star(t)$*. Moreover, assume that the latest prediction* $\bar{\chi}(\Delta t|t)$ *is contained in the region of attraction of* $\bar{\chi}^\star(t+\Delta t)$*. Then we get:*

$$\begin{aligned} \Delta\bar{\chi}(t+\Delta t) &= \Delta\bar{\chi}(\Delta t|t) - H^{-1}(t)\nabla\tilde{J}(t+\Delta t) - (\bar{\chi}^\star(t+\Delta t) - \bar{\chi}^\star(\Delta t|t)) \\ &= \Delta\bar{\chi}(t) - H^{-1}(t)\left(\nabla\tilde{J}'(t+\Delta t) - \frac{\partial \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}}(\xi(\Delta t|t), \bar{\chi}^\star(\Delta t|t))\right) \\ &\quad + O(\|e(t+\Delta t)\|), \end{aligned} \tag{94}$$

where $\limsup_{\eta\to 0} \frac{O(\eta^2)}{\eta^2} < \infty$*,* $\bar{\chi}^\star(\Delta t|t)$ *is a local minimizer associated with* $\xi(\Delta t|t)$ *and* $\Delta\bar{\chi}(\Delta t|t) = \Delta\bar{\chi}(t)$*. The extension of the right-hand side in the second step is feasible as we add zero. Due to Theorem 4.2, we can also overbound the difference* $(\bar{\chi}^\star(t+\Delta t) - \bar{\chi}^\star(\Delta t|t))$*.*

Now we use the fact that $\tilde{J}'_{\bar{\theta}(t)}(x(t), \bar{\chi}(t))$ *is twice continuously differentiable and its partial derivatives are locally Lipschitz continuous in order to arrive at*

$$\begin{aligned} &\nabla\tilde{J}'(t+\Delta t) - \frac{\partial \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}}(\xi(\Delta t|t), \bar{\chi}^\star(\Delta t|t)) \\ = \; &\frac{\partial^2 \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}^2}(\xi(\Delta t|t), \bar{\chi}^\star(\Delta t|t))\Delta\bar{\chi}(\Delta t|t) + \frac{\partial^2 \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}\partial\xi}(\xi(\Delta t|t), \bar{\chi}^\star(\Delta t|t))e(t+\Delta t) \\ &+ O(||e(t+\Delta t)||^2) + O(\|e(t+\Delta t)\|\|\Delta\bar{\chi}(\Delta t|t)\|) + O(\|\Delta\bar{\chi}(\Delta t|t)\|^2). \end{aligned} \tag{95}$$

Plugging (95) *into* (94) *yields*

$$\begin{aligned} \Delta\bar{\chi}(t+\Delta t) &= -H^{-1}(t)\frac{\partial^2 \tilde{J}'_{\bar{\theta}(t)}}{\partial \bar{\chi}\partial\xi}(\xi(\Delta t|t), \bar{\chi}^\star(\Delta t|t))e(t+\Delta t) \\ &\quad + O(\|e(t+\Delta t)\|) + O(\|e(t+\Delta t)\|\|\Delta\bar{\chi}(\Delta t|t)\|) + O(\|\Delta\bar{\chi}(\Delta t|t)\|^2), \end{aligned} \tag{96}$$

and thus there exists a bounded constant $K > 0$ *such that*

$$||\bar{\chi}(t+\Delta t) - \bar{\chi}^\star(t+\Delta t|| \leq K\left(||\Delta\bar{\chi}(t)||^2 + ||e(t+\Delta t)||\right),$$

which finishes the proof.

The convergence rate of Algorithm 4.1 is linear in $\|e(t)\|$ and quadratic in the difference $\|\bar{\chi}(t) - \bar{\chi}^\star(t)\|$ from a local optimizer. This result is not too surprising, since we are updating the activation durations with the Newton direction. One can compare this convergence analysis with the one we obtained for Algorithm 3.1 (equation (30)).

4.4 Conclusions

We introduced in this chapter a novel optimal adaptive feedback control scheme for nonlinear autonomous switched systems under the influence of unknown disturbances. Mode transitions are triggered by the controller at the intersections of the state trajectory with a time-varying switching plane. The plane itself is adjusted in real-time by a switching plane controller to produce an overall loop behavior, which is optimal with respect to a user-defined performance metric.

A central benefit of the proposed cascaded loop structure concerns the fact that as the first of its kind, it unifies two feedback concepts for this class of systems. As a direct consequence of the latter, this optimal adaptive feedback control strategy produces a behavior, which is more robust in presence of small and fast disturbances than other approaches proposed in this thesis. However, we also showed this class of explicit feedback control approaches can only be applied to free final-time optimal control problems.

CHAPTER V

OPTIMAL TIMING CONTROL WITH CHANGING COST FUNCTION

In many applications that operate in real-time, the environment of the system changes in the form of the cost function, meaning that the cost function may be a function of the current system time t. To be precise, the instantaneous trajectory cost function $L(x, t)$ may not be known a priori, but rather the system learns about it incrementally as time advances (here, t denotes the time index for the real-time operation). This is for instance the case when the system is tracking a target with unknown dynamics. In this case, we consider the problem of optimizing a cost-to-go performance functional defined by a time varying cost function, where the cost function is updated on-line.

This problem represents a new class of real-time optimal control problems, where new information is injected into the problem on-line, and we do not know a priori how the cost function varies when the switched system is in operation. We show that the information required to update the switching times is the 'rate of change' of the instantaneous cost, which is in the form of the partial derivative of the instantaneous cost function. This rate of change is not known before operation of the plant, thus a real-time framework must be used to adopt to these new information.

In this chapter, we first obtain update differential equation for the optimal switching times, using the real-time information. This differential equation provides a continuous process to update the switching times. Then, based on the continuous process, we propose real-time algorithms that adaptively adjusts the estimated value of the optimal switching time vector as new information about the cost function becomes available. In other words, it attempts to track the solution point of the optimal cost-to-go problem as it learns about changes in the cost function.

5.1 Continuous Process

Similar to Chapter 3, we first propose a continuous process, which corresponds to a differential equation that describes the evolution of the optimal switching time vector. The differential equation describes how the optimal switching time vector evolves with respect to the varying cost function.

Again, consider an autonomous switched system

$$\dot{x} = f_i(x), t \in [\tau_{i-1}, \tau_i), i = 1, 2, ..., N+1, \tag{97}$$

where $x(t) \in \mathbb{R}^n$ is the state variable, initial condition $x(0) = x_0$ and final time T are given. Modal functions $f_i : \mathbb{R}^n \to \mathbb{R}^n$ correspond to a fixed mode sequence $\bar{q} = \{0, 1, ..., N\}\}$ and the notations $\tau_0 = 0$, $\tau_{N+1} = T$ are used. We assume that the state of the system is known (the real-time optimal control problem with an observer was addressed in Chapter 3). The control input of this system is the switching time vector $\bar{\tau} = [\tau_1, \tau_2, ..., \tau_N]^T$, where it must satisfy the constraint $\bar{\tau} \in \Lambda$ and

$$\Lambda := \{\bar{\tau} = [\tau_1, ..., \tau_N]^T : 0 = \tau_0 \leq \tau_1 \leq ... \leq \tau_N \leq \tau_{N+1} = T\}. \tag{98}$$

5.1.1 The Single-switch Case

To capture the essence of our method, let us first focus on the case where there is only one switching, hence $N = 1$. Therefore, we can simplify the notation by deleting the subscript from τ_1, and denote the switching time by τ. In this case, according to (97), $\dot{x} = f_1(x)$ for $t \in [0, \tau)$, and $\dot{x} = f_2(x)$ for $t \in [\tau, T]$. However, we have to be careful about notation here, since t is used as the time index for the real-time execution of the system. In light of this, a simple translation of Equation (97) (with $N = 1$) that emphasizes the dependence of the switching time on t may not be well defined, and we have to carefully define the state trajectory $x(t)$ and the switching time $\tau(t)$ simultaneously.

Since we consider dynamic computations of the switching time, we denote the switching time vector by $\tau(t)$ to emphasize its dependence on t. Furthermore, we start by defining $\tau(t)$ to be a minimum point of the cost-to-go (defined below), and since this may vary continuously with t, we first describe $\{\tau(t)\}_{t=0}^{T}$ as a continuous process; our algorithms will be based on it and they will perform their computations at a discrete set of times. To this end, it is convenient to first define the state trajectory of the dynamics associated with the cost-to-go performance measure.

Given $x \in \mathbb{R}^n$, $\tau \in [0,T]$, and $t \in [0,T]$, define $\tilde{x}(s,t,x,\tau)$, $s \geq t$, to be the state trajectory simulated forward from t, as a function of s, with the initial condition of x at the time $s = t$; that is, $\tilde{x}(s,t,x,\tau)$ is defined via the equation

$$\frac{\partial \tilde{x}}{\partial s} = \begin{cases} f_1(\tilde{x}), & \text{if } s \leq \tau \\ f_2(\tilde{x}), & \text{if } s > \tau, \end{cases}$$

with the initial condition $\tilde{x}(t,t,x,\tau) = x$. We denote $\tilde{x}$ as the predicted (or simulated) state trajectory starting at time $s = t$ and it evolves until $s = T$.

Given $t \in [0,T]$, let $L(x,s,t)$ be a cost function of $x \in \mathbb{R}^n$ and future times $s \geq t$. It has the following meaning: At time t the system has "learned" about the cost function of the state variable x and the future times s, and this function is $L(x,s,t)$. For example, if it is desired to track a moving target whose position and velocity are known at time t, $L(x,s,t)$ may indicate the square-distance of the state (x) from the goal's projected position at future times (s) based on its position and velocity at time t. Generally, the cost-to-go performance measure consists of the function $L(x,s,t)$, and we assume that it is continuous, continuously differentiable in x for every $t \in [0,T]$ and $s \in [t,T]$, and piecewise continuously differentiable in t and s for every $x \in \mathbb{R}^n$. Lastly, we assume that $\frac{\partial L}{\partial x}(x,s,t)$ to be differentiable in t (we will show that this partial derivative represent the required rate of change information). These assumptions are required for the continuous process, but not for the discrete algorithms.

For every $\tau \in [0,T]$, $t \in [0,T]$, and $x \in \mathbb{R}^n$, we define the cost-to-go performance functional, $J(t,x,\tau)$, by

$$J(t,x,\tau) \;=\; \int_t^T L\big(\tilde{x}(s,t,x,\tau),s,t\big)ds. \tag{99}$$

We define $\tau(t)$ as a local minimum point of the cost-to-go (99) from time t and the initial state $x(t)$ at that time, where $x(t)$ is the trajectory of the system. As mentioned earlier, we have to formally define the processes $\{\tau(t)\}$ and $\{x(t)\}$ simultaneously, and this is done in the following way: $x(t)$ satisfies the differential equation

$$\frac{dx}{dt} \;=\; \begin{cases} f_1(x(t)), & \text{if } t \leq \tau(t) \\ f_2(x(t)), & \text{if } t > \tau(t) \end{cases}$$

with a given boundary condition $x(0) = x_0$; and as for $\tau(t)$, as long as $t < \tau(t)$ then $\tau(t)$ is defined as a function that satisfies

$$\frac{\partial J}{\partial \tau}\big(t, x(t), \tau(t)\big) \;=\; 0, \tag{100}$$

with a given boundary condition $\tau(0) = \tau_0 \in [0,T]$ such that $\frac{\partial J}{\partial \tau}\big(0, x(0), \tau_0\big) \;=\; 0$. If $t \geq \tau(t)$, we define $\dot{\tau}(t) = 0$. The condition $\dot{\tau}(t) = 0$ when $\tau(t) \geq t$ reflects the fact that once the switching time has occurred it become a part of the past and cannot be modified, in other words, once $t \geq \tau(t)$, $\tau(\xi)$ has a constant value $\forall\ \xi \geq t$. Similarly, after the switching, the gradient $\frac{\partial J}{\partial \tau}\big(t, x(t), \tau(t)\big)$ is set to 0 simply because the switching time can no longer be modified.

To help clarify the setting and analysis of this problem, refer to Figure 19 where the state x and projected state $\tilde{x}$ are shown for time t and $t + \Delta t$. The goal of this section is to obtain a differential equation for $\tau(t)$ so that τ is locally optimal with respect to the cost-to-go defined in (99) and thus (100) is satisfied. We aim to provide a method that is more computationally feasible than solving a sequence of optimization problem for the cost-to-go functional (99).

Throughout this section, we assume that the processes $\{x(t)\}$ and $\{\tau(t)\}$ are well defined in the above manner and $\tau(t)$ is continuous in t. We should note that, this

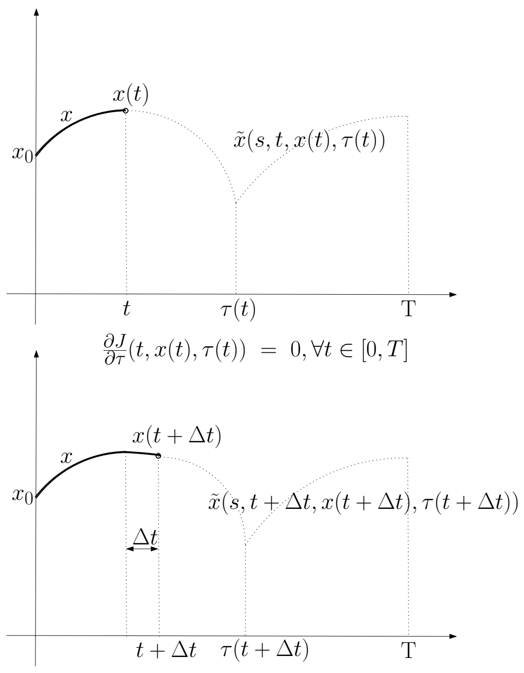

$x(t)$
x
x_0
$\tilde{x}(s,t,x(t),\tau(t))$
t
$\tau(t)$
T
$\frac{\partial J}{\partial \tau}(t,x(t),\tau(t)) = 0, \forall t \in [0,T]$
$x(t+\Delta t)$
x
x_0
$\tilde{x}(s,t+\Delta t,x(t+\Delta t),\tau(t+\Delta t))$
Δt
$t+\Delta t$
$\tau(t+\Delta t)$
T

Figure 19: The state x, projected state $\tilde{x}$ are shown at time t and time $t+\Delta t$. The goal is to provide an update differential equation for $\tau(t)$ so that it stays optimal with respect to the cost-to-go (99).

assumption may not always hold for all problems, since extrema are known to not always be continuous across varying parameters of the optimization problem. However, again this assumption is no longer required when we implement this continuous process in the form of a discrete algorithm, because the switching times are updated in a discrete manner.

Consider a time-point t such that $t < \tau(t)$. Since $\frac{\partial J}{\partial \tau}(t,x(t),\tau) = 0$, taking the total derivative with respect to t, it follows that

$$\frac{d}{dt}\frac{\partial J}{\partial \tau}\big(t,x(t),\tau(t)\big) \;=\; 0$$

as well. This total derivative has the following form,

$$\frac{\partial^2 J}{\partial t \partial \tau}\big(t,x(t),\tau(t)\big) + \frac{\partial^2 J}{\partial x \partial \tau}\big(t,x(t),\tau(t)\big)\dot{x}(t) + \frac{\partial^2 J}{\partial \tau^2}\big(t,x(t),\tau(t)\big)\dot{\tau}(t) \;=\; 0,$$

and hence,

$$\dot{\tau}(t) \ = -\Big(\frac{\partial^2 J}{\partial \tau^2}\big(t, x(t), \tau(t)\big)\Big)^{-1}\Big(\frac{\partial^2 J}{\partial t \partial \tau}\big(t, x(t), \tau(t)\big) + \frac{\partial^2 J}{\partial x \partial \tau}\big(t, x(t), \tau(t)\big)\dot{x}(t)\Big). \tag{101}$$

By the assumption that $\tau(t)$ is well defined and continuous, the second derivative $\frac{\partial^2 J}{\partial \tau^2}\big(t, x(t), \tau(t)\big)$ is positive. We next simplify the right-hand side of Equation (101).

Define the function $\phi_t(\xi)$, $\xi \geq t$, by

$$\phi_t(\xi) \ = \ \frac{\partial J}{\partial \tau}\big(\xi, x(\xi), \tau(t)\big).$$

Then

$$\frac{d\phi_t}{d\xi}(\xi) \ = \ \frac{\partial^2 J}{\partial t \partial \tau}\big(\xi, x(\xi), \tau(t)\big) + \frac{\partial^2 J}{\partial x \partial \tau}\big(\xi, x(\xi), \tau(t)\big)\dot{x}(\xi), \tag{102}$$

and at $\xi = t$,

$$\frac{d\phi_t}{d\xi}(t) \ = \ \frac{\partial^2 J}{\partial t \partial \tau}\big(t, x(t), \tau(t)\big) + \frac{\partial^2 J}{\partial x \partial \tau}\big(t, x(t), \tau(t)\big)\dot{x}(t). \tag{103}$$

Plug this in (101) to obtain,

$$\dot{\tau}(t) \ = \ -\Big(\frac{\partial^2 J}{\partial \tau^2}\big(t, x(t), \tau(t)\big)\Big)^{-1}\frac{d\phi_t}{d\xi}(t). \tag{104}$$

Let us next derive an expression for $\frac{d\phi_t}{d\xi}(t)$.

Given $t \in [0, T]$ and $x \in \mathbb{R}^n$, denote by $\Phi_{t,x}(s, s_0)$ the state transition matrix associated with the matrix-valued function (of s) $-\frac{\partial f_2}{\partial x}\big(\tilde{x}(s, t, x, \tau)\big)^T$. Thus,

$$\frac{\partial}{\partial s}\Phi_{t,x}(s, s_0) \ = \ -\frac{\partial f_2}{\partial x}\big(\tilde{x}(s, t, x, \tau)\big)^T \Phi_{t,x}(s, s_0), \tag{105}$$

with $\Phi_{t,x}(s_0, s_0) = I$ (the $n \times n$ identity matrix).

Given $t \in [0, T)$ and $\Delta T > 0$ such that $t + \Delta t \leq T$.

Lemma 5.1 *For all $s \geq t + \Delta t$ and for all $s_0 \geq t + \Delta t$,*

$$\Phi_{t+\Delta t, \tilde{x}(t+\Delta t, t, x, \tau)}(s, s_0) \ = \ \Phi_{t,x}(s, s_0). \tag{106}$$

Proof 5.1 *Fix $t \in [0,T)$ and $\Delta t > 0$ such that $t+\Delta t \leq T$. $\forall\ s \in [t+\Delta t, T]$ and $\forall\ s_0 \in [t+\Delta t, T]$, equation (105) is in force. Moreover, the same equation holds with $t+\Delta t$ and $\tilde{x}(t+\Delta t, t, x, \tau)$ in lieu of t and x, respectively, meaning that*

$$\frac{\partial}{\partial s}\Phi_{t+\Delta t, \tilde{x}(t+\Delta t,t,x,\tau)}(s,s_0) = -\Big(\frac{\partial f_2}{\partial x}\big(\tilde{x}(s, t+\Delta t, \tilde{x}(t+\Delta t, t, x, \tau), \tau)\big)\Big)^T \Phi_{t+\Delta t, \tilde{x}(t+\Delta t,t,x,\tau)}(s,s_0). \tag{107}$$

Now $\forall\ s \geq t+\Delta t$,

$$\frac{\partial \tilde{x}}{\partial s}(s,t,x,\tau) = \begin{cases} f_1(\tilde{x}), & \text{if } s \leq \tau \\ f_2(\tilde{x}), & \text{if } s > \tau \end{cases}$$

with the boundary condition $\tilde{x}(t,t,x,\tau) = x$, and

$$\frac{\partial \tilde{x}}{\partial s}\big(s, t+\Delta t, \tilde{x}(t+\Delta t, t, x, \tau), \tau\big) = \begin{cases} f_1(\tilde{x}), & \text{if } s \leq \tau \\ f_2(\tilde{x}), & \text{if } s \geq \tau \end{cases}$$

with the boundary condition $\tilde{x}(t+\Delta t, t+\Delta t, \tilde{x}(t+\Delta t, t, x, \tau), \tau) = \tilde{x}(t+\Delta t, t, x, \tau)$. We see that $\tilde{x}(s,t,x,\tau)$ and $\tilde{x}(s, t+\Delta t, \tilde{x}(t+\Delta t, t, x, \tau), \tau)$ satisfy the same differential equation (in s) $\forall\ s \geq t+\Delta t$, and these two functions have the same value at $s = t+\Delta t$; consequently

$$\tilde{x}(s,t,x,\tau) = \tilde{x}(s, t+\Delta t, \tilde{x}(t+\Delta t, t, x, \tau), \tau) \tag{108}$$

for all $s \in [t+\Delta t, T]$. Therefore, by (105) and (107), $\Phi_{t+\Delta t, \tilde{x}(t+\Delta t,t,x,\tau)}(s,s_0)$ and $\Phi_{t,x}(s,s_0)$ satisfy the same differential equation (in s), with the same value at $s = s_0$ (i.e., I), and this implies (106).

We now can characterize the rightmost term in the right-hand side of (104), and hence obtain a computable expression for $\dot{\tau}(t)$.

Theorem 5.1 *Suppose that $t < \tau(t)$. Then the derivative term $\frac{d\phi_t}{d\xi}(\xi)$ at $\xi = t$ has the following form,*

$$\frac{d\phi_t}{d\xi}(t) = \int_{\tau(t)}^{T} \frac{\partial^2 L}{\partial t \partial x}\big(\tilde{x}(s,t,x(t),\tau(t)), s, t\big)\ \big(\Phi_{t,x(t)}(\tau(t),s)\big)^T ds \Big(f_1\big(\tilde{x}(\tau(t),t,x(t),\tau(t))\big) - f_2\big(\tilde{x}(\tau(t),t,x(t),\tau(t))\big)\Big). \tag{109}$$

Proof 5.2 *Fix $t \in [0,T)$ and $x \in \mathbb{R}^n$, and consider $J(x,t,\tau)$ as a function of $\tau > t$. Define the costate $p(s) \in \mathbb{R}^n$, $s \in [\tau, T]$, via the following differential equation,*

$$\dot{p}(s) = -\Big(\frac{\partial f_2}{\partial x}\big(\tilde{x}(s,t,x,\tau)\big)\Big)^T p(s) - \Big(\frac{\partial L}{\partial x}\big(\tilde{x}(s,t,x,\tau),s,t\big)\Big)^T$$

with the boundary condition $p(T) = 0$. By Equation (105),

$$p(s) = \int_s^T \Phi_{t,x}(s,\xi)\frac{\partial L}{\partial x}\big(\tilde{x}(\xi,t,x,\tau),\xi,t\big)^T d\xi, \tag{110}$$

and at $s = \tau$, and replacing ξ by s in (110), we obtain,

$$p(\tau) = \int_\tau^T \Phi_{t,x}(\tau,s)\frac{\partial L}{\partial x}\big(\tilde{x}(s,t,x,\tau),s,t\big)^T ds. \tag{111}$$

Next, by using the gradient formula established in Chapter 2 (see equation (13))

$$\frac{\partial J}{\partial \tau}(t,x,\tau) = p(\tau)^T\Big(f_1\big(\tilde{x}(\tau,t,x,\tau)\big) - f_2\big(\tilde{x}(\tau,t,x,\tau)\big)\Big), \tag{112}$$

and hence, and by (111),

$$\begin{aligned}\frac{\partial J}{\partial \tau}(t,x,\tau) &= \int_\tau^T \frac{\partial L}{\partial x}\big(\tilde{x}(s,t,x,\tau),s,t\big)\Big(\Phi_{t,x}(\tau,s)\Big)^T ds \\ &\quad \Big(f_1\big(\tilde{x}(\tau,t,x,\tau)\big) - f_2\big(\tilde{x}(\tau,t,x,\tau)\big)\Big).\end{aligned} \tag{113}$$

Use (113) with two specific values: $(t, x(t), \tau(t))$, and $(t+\Delta t, x(t+\Delta t), \tau(t))$, to obtain the following two equations,

$$\begin{aligned}\frac{\partial J}{\partial \tau}(t,x(t),\tau(t)) &= \int_{\tau(t)}^T \frac{\partial L}{\partial x}\big(\tilde{x}(s,t,x(t),\tau(t)),s,t\big)\Big(\Phi_{t,x(t)}(\tau(t),s)\Big)^T ds \\ &\times \Big(f_1\big(\tilde{x}(\tau(t),t,x(t),\tau(t))\big) - f_2\big(\tilde{x}(\tau(t),t,x(t),\tau(t))\big)\Big),\end{aligned} \tag{114}$$

and

$$\begin{aligned}&\frac{\partial J}{\partial \tau}(t+\Delta t, x(t+\Delta t,\tau(t)) \\ = &\int_{\tau(t)}^T \frac{\partial L}{\partial x}\big(\tilde{x}(s,t+\Delta t,x(t+\Delta t),\tau(t)),s,t+\Delta t\big)\Big(\Phi_{t+\Delta t,x(t+\Delta t)}(\tau(t),s)\Big)^T ds \\ &\Big(f_1\big(\tilde{x}(\tau(t),t+\Delta t,x(t+\Delta t),\tau(t))\big) - f_2\big(\tilde{x}(\tau(t),t+\Delta t,x(t+\Delta t),\tau(t))\big)\Big).\end{aligned} \tag{115}$$

Now suppose that $t+\Delta t < \tau(t)$. *Then* $\forall \xi \in [t, t+\Delta t]$, $\frac{dx}{d\xi} = f_1(x(\xi))$ *and* $\frac{\partial \tilde{x}}{\partial \xi}\big(\xi, t, x(t), \tau(t)\big) = f_1\big(\tilde{x}(\xi, t, x(t), \tau(t))\big)$ *with* $\tilde{x}(t, t, x(t), \tau(t)) = x(t)$, *and hence,* $\forall \xi \in [t, t+\Delta t]$,

$$\tilde{x}(\xi, t, x(t), \tau(t)) \;=\; x(\xi).$$

In particular, for $\xi = t + \Delta t$,

$$\tilde{x}(t+\Delta t, t, x(t), \tau(t)) \;=\; x(t+\Delta t). \qquad (116)$$

By Lemma 5.1 and Equation (116), $\forall\ s \geq t+\Delta t$ *and* $\forall\ s_0 \geq t + \Delta t$,

$$\Phi_{t+\Delta t, x(t+\Delta t)}(s, s_0) \;=\; \Phi_{t, x(t)}(s, s_0).$$

Therefore, and by (115),

$$\begin{aligned} &\frac{\partial J}{\partial \tau}(t+\Delta t, x(t+\Delta t), \tau(t)\big) \\ = \;& \int_{\tau(t)}^{T} \frac{\partial L}{\partial x}\big(\tilde{x}(x, t+\Delta t, x(t+\Delta t), \tau(t)\big), s, t+\Delta t\big)\Big(\Phi_{t,x(t)}(\tau(t), s)\Big)^T ds \\ & \Big(f_1\big(\tilde{x}(\tau(t), t+\Delta t, x(t+\Delta t), \tau(t))\big) - f_2\big(\tilde{x}(\tau(t), t+\Delta t, x(t+\Delta t), \tau(t))\big)\Big). \end{aligned} \qquad (117)$$

Moreover, $\forall s \in [t+\Delta t, \tau(t)]$, $\frac{\partial}{\partial s}\tilde{x}(s, t+\Delta t, x(t+\Delta t), \tau(t)) = f_1\big(\tilde{x}(s, t+\Delta t, x(t+\Delta t), \tau(t)\big)$ *and* $\frac{\partial}{\partial s}\tilde{x}(s, t, x(t), \tau(t)) = f_1\big(\tilde{x}(s, t, x(t), \tau(t))\big)$; *while the initial conditions of the respective terms in these two differentiable equations are* $\tilde{x}(t+\Delta t, t+\Delta t, x(t+\Delta t), \tau(t)) = x(t+\Delta t)$ *by definition, and* $\tilde{x}(t+\Delta t, t, x(t), \tau(t)) = x(t+\Delta t)$ *by Equation (114). Consequently,* $\forall s \in [t+\Delta t, \tau(t)]$,

$$\tilde{x}(s, t+\Delta t, x(t+\Delta t), \tau(t)) \;=\; \tilde{x}(s, t, x(t), \tau(t)).$$

Using this in (117) we obtain,

$$\begin{aligned} &\frac{\partial J}{\partial \tau}\big(t+\Delta t, x(t+\Delta t), \tau(t)\big) \;= \\ &\int_{\tau(t)}^{T} \frac{\partial L}{\partial x}\big(\tilde{x}(s, t, x(t), \tau(t)), s, t+\Delta t\big)\Big(\Phi_{t,x(t)}(\tau(t), s)\Big)^T ds \\ &\Big(f_1\big(\tilde{x}(\tau(t), t, x(t), \tau(t))\big) - f_2\big(\tilde{x}(\tau(t), t, x(t), \tau(t))\big)\Big). \end{aligned} \qquad (118)$$

By (118) and (114),

$$\begin{aligned}
&\frac{\partial J}{\partial \tau}\big(t+\Delta t, x(t+\Delta t), \tau(t)\big) - \frac{\partial J}{\partial \tau}\big(t, x(t), \tau(t)\big) \\
= \quad & \int_{\tau(t)}^{T} \Big[\Big(\frac{\partial L}{\partial x}\big(\tilde{x}(s,t,x(t),\tau(t)), s, t+\Delta t\big) - \frac{\partial L}{\partial x}\big(\tilde{x}(s,t,x(t),\tau(t)) s, t\big)\Big) \\
& \Big(\Phi_{t,x(t)}(\tau(t), s)\Big)^T\Big] ds \ \Big(f_1\big(\tilde{x}(\tau(t),t,x(t),\tau(t)\big) - f_2\big(\tilde{x}(\tau(t),t,x(t),\tau(t))\big)\Big).
\end{aligned} \tag{119}$$

Dividing both sides of (119) by Δt, taking the limit $\Delta t \to 0$, and recalling the definition of $\phi_t(\xi)$, equation (109) follows.

Finally, we obtain an expression for $\dot{\tau}(t)$.

Theorem 5.2 *For every $t \in [0,T)$ such that $t < \tau(t)$, we have that*

$$\begin{aligned}
&\dot{\tau}(t) \\
= \quad & -\Big(\frac{\partial^2 J}{\partial \tau^2}\big(t, x(t), \tau(t)\big)\Big)^{-1} \int_{\tau(t)}^{T} \frac{\partial^2 L}{\partial t \partial x}\big(\tilde{x}(s,t,x(t),\tau(t)), s, t\big) \Big(\Phi_{t,x(t)}(\tau(t), s)\Big)^T ds \\
& \Big(f_1\big(\tilde{x}(\tau(t),t,x(t),\tau(t))\big) - f_2\big(\tilde{x}(\tau(t),t,x(t),\tau(t))\big)\Big).
\end{aligned} \tag{120}$$

Proof 5.3 *Follows directly from (103), (104) and Theorem 5.1.*

Some remarks are in order. First, the term $\frac{\partial^2 L}{\partial t \partial x}\big(\tilde{x}(s,t,x(t),\tau(t)), s, t\big)$ should be interpreted as the *rate of change* of the instantaneous cost function. This information is required to update the switching time optimally.

Second, if the rate of change $\frac{\partial^2 L}{\partial t \partial x}\big(\tilde{x}(s,t,x(t),\tau(t)), s, t\big)$ does not depend on s, then the computation of the gradient is much easier since $\frac{\partial^2 L}{\partial t \partial x}\big(\tilde{x}(s,t,x(t),\tau(t)), s, t\big)$ can be pulled outside of the integration. Notice that in this case, the integration can be completely performed off-line. This simplification can be done in many cases, for example when one attempts to drive the trajectory of the switched system towards a time-varying signal $x_c(t)$ ($L = ||\tilde{x}(s) - x_c(t)||^2$).

Third, instead of solving a sequence of optimization problem to minimize the cost-to-go functional (99), the iteration process presented in this chapter requires obtaining the second derivative and integration forward only once.

Last, the switching time evolution $\dot{\tau}(t)$ is only valid for time $t \in [0, \tau(t)]$, and the mode transition is triggered at time $\tau(t)$. After this time, i.e. $t \in (\tau(t), T]$, the switching time is considered "frozen" since the switch has already happened.

5.1.2 The Case of Multiple Switching Times

Consider the case where there are N switching times, τ_i, $i = 1, \ldots, T$, denoted collectively by the vector $\bar{\tau} := [\tau_1, \ldots, \tau_N]^T \in \mathbb{R}^N$. We assume that $\bar{\tau} \in \Lambda$. For a fixed vector $\bar{\tau}$, the state equation is given via (97), but since $\bar{\tau}$ is a function of time, it is denoted by $\bar{\tau}(t)$, and we define it together with the state trajectory as it was done for the single-switching case.

Given a switching vector $\bar{\tau} \in \Lambda$, $x \in \mathbb{R}^n$, $t \in [\tau_0, T]$, and $s \in [t, T]$, define $\tilde{x}(s, t, x, \bar{\tau}), s \geq t$ in a piecewise manner, via the following equations:

$$\frac{\partial \tilde{x}}{\partial s} = f_i(\tilde{x}(s)), s \in (\tau_i, \tau_{i+1}], i = 1, 2, .., N+1, \tag{121}$$

where f_1 are modal functions in the corresponding mode sequence and with initial condition $\tilde{x}(t, t, x, \bar{\tau}) = x$.

Similar to the previous section, let $L(x, s, t)$ be a cost function, $s \geq t$. Define $J(t, x, \tau)$ by

$$J(t, x, \bar{\tau}) \ = \ \int_t^T L\big(\tilde{x}(s, t, x, \bar{\tau}), s, t\big) ds. \tag{122}$$

Define $x(t)$ and $\bar{\tau}(t) = [\tau_1(t), ..., \tau_N(t)]$ simultaneously in the following way: $x(t)$ satisfies the differential equation

$$\frac{dx}{dt} = f_i(x(t)), t \in (\tau_i(t), \tau_{i+1}(t)], i = 1, 2, .., N, \tag{123}$$

with a given boundary condition $x(0) = x_0$; and as for $\bar{\tau}(t)$, as long as $t < \tau_1(t)$ then

$$\frac{\partial J}{\partial \bar{\tau}}\big(t, x(t), \bar{\tau}(t)\big) \ = \ 0 \tag{124}$$

with a given boundary condition $\bar{\tau}(0) = \bar{\tau}_0$. Furthermore, we assume that $\bar{\tau}(t)$ is continuous in t.

Define the function $\phi_t(\xi)$, $\xi \geq t$, by

$$\phi_{i,t}(\xi) \ = \begin{bmatrix} \phi_{1,t}(\xi) \\ \phi_{2,t}(\xi) \\ \vdots \\ \phi_{N,t}(\xi) \end{bmatrix}, \tag{125}$$

where $\phi_{i,t}(\xi)$ is defined as

$$\phi_{i,t}(\xi) = \frac{\partial J}{\partial \tau_i}\big(\xi, x(\xi), \bar{\tau}(t)\big) \tag{126}$$

With the exact same mathematical equations, if $\bar{\tau}(t) \in \Lambda$, then the evolution of $\bar{\tau}(t)$ is

$$\dot{\bar{\tau}}(t) \ = -\Big(\frac{\partial^2 J}{\partial \tau^2}\big(t, x(t), \bar{\tau}(t)\big)\Big)^{-1}\Big(\frac{\partial^2 J}{\partial t \partial \tau}\big(t, x(t), \bar{\tau}(t)\big) + \frac{\partial^2 J}{\partial x \partial \tau}\big(t, x(t), \bar{\tau}(t)\big)\dot{x}(t)\Big), \tag{127}$$

for $t < \tau_1(t)$, and

$$\dot{\bar{\tau}}(t) \ = \ -\Big(\frac{\partial^2 J}{\partial \bar{\tau}^2}\big(t, x(t), \bar{\tau}(t)\big)\Big)^{-1}\frac{d\phi_t}{d\xi}(t). \tag{128}$$

Note that $\frac{d\phi_t}{d\xi}(t)$ is now a vector in $\mathbb{R}^N$.

Given t and $x \in R^n$, denote by $\Phi_{i,t,x}(s, s_0)$ the state transition matrix associated with the matrix-valued function (of s) $-\frac{\partial f_i}{\partial x}\big(\tilde{x}(s, t, x, \bar{\tau})\big)^T$. Thus,

$$\frac{\partial}{\partial s}\Phi_{i,t,x}(s, s_0) \ = \ -\frac{\partial f_i}{\partial x}\big(\tilde{x}(s, t, x, \bar{\tau})\big)^T\Phi_{i,t,x}(s, s_0), i = 2, ..., N. \tag{129}$$

with $\Phi_{i,t,x}(s_0, s_0) = I$ (the identity matrix).

Given $t \in [0, T)$ and $\Delta t > 0$ such that $t + \Delta t \leq T$.

Lemma 5.2 *$\forall$ $s \geq t + \Delta t$ and $\forall$ $s_0 \geq t + \Delta t$,*

$$\Phi_{i,t+\Delta t,\tilde{x}(t+\Delta t,t,x,\bar{\tau})}(s, s_0) \ = \ \Phi_{i,t,x}(s, s_0), \forall i = 1, 2, ..., N. \tag{130}$$

Furthermore, $\forall s \in [t + \Delta t, T]$,

$$\tilde{x}(s, t + \Delta t, x(t + \Delta t), \bar{\tau}(t)) \ = \ \tilde{x}(s, t, x(t), \bar{\tau}(t)). \tag{131}$$

Proof 5.4 *Follows using the exact same argument as in Lemma 5.1.*

Next we derive the equations to obtain $\frac{d\phi_t}{d\xi}(t)$. Before this results in presented, define the costate trajectory $p(s,t,x,\bar{\tau}), s \in [\tau_1, T]$ corresponding to the trajectory $\tilde{x}(s,t,x,\bar{\tau})$ via the following differential equation

$$\frac{\partial p}{\partial s}(s,t,x,\bar{\tau}) = -\Big(\frac{\partial f_i}{\partial x}\big(\tilde{x}(s,t,x,\bar{\tau})\big)\Big)^T p(s) - \Big(\frac{\partial L}{\partial x}\big(\tilde{x}(s,t,x,\bar{\tau}),s,t\big)\Big)^T, \qquad (132)$$

for $s \in [\tau_{i-1}, \tau_i), i = 2, ..., N$ and the boundary condition $p(T,t,x,\bar{\tau}) = 0$. Using the state transition matrices, the costate trajectory can be obtained (in a backward direction) as

$$\begin{aligned} p(\tau_i,t,x,\bar{\tau}) &= \Phi_{i,t,x}(\tau_i,\tau_{i+1})p(\tau_{i+1},t,x,\tau) + \int_{\tau_i}^{\tau_{i+1}} \Phi_{i,t,x}(\tau,s)\frac{\partial L}{\partial x}\big(\tilde{x}(s,t,x,\bar{\tau}),s,t\big)^T ds, \\ & i = 1, ..., N. \qquad (133) \end{aligned}$$

Notice that $p(\tau_{N+1},t,x,\bar{\tau}) = 0$. Lastly, if we plug-in ξ, $x(\xi)$ and $\bar{\tau}(t)$ in places of t and x and $\bar{\tau}$ in the definition of $\tilde{x}$, we have $\tilde{x}(s,\xi,x(\xi),\bar{\tau}(t))$ and its corresponding costate $p(s,\xi,x(\xi),\bar{\tau}(t))$. Define

$$\psi_{i,t}(\xi) = p(\tau_i(t),\xi,x(\xi),\bar{\tau}(t)), \qquad (134)$$

then we can show the following theorem.

Theorem 5.3 *Suppose that* $t < \tau_1(t)$. *Then the derivative term* $\frac{d\phi_t}{d\xi}(\xi) = [\frac{d\phi_{1,t}}{d\xi}(\xi), ..., \frac{d\phi_{N,t}}{d\xi}(\xi)]^T$ *at* $\xi = t$ *has the following form*

$$\begin{aligned} \frac{d\phi_{i,t}}{d\xi}(t) &= \Bigg(\Big(\frac{d\psi_{i+1,t}}{d\xi}(t)\Big)^T \Big(\Phi_{i,t,x(t)}(\tau_i(t),\tau_{i+1}(t))\Big)^T \\ &\quad + \int_{\tau_i(t)}^{\tau_{i+1}(t)} \frac{\partial^2 L}{\partial t \partial x}\big(\tilde{x}(s,t,x(t),\bar{\tau}(t)),s,t\big)\left(\Phi_{i,t,x(t)}(\tau_{i+1}(t),s)\right)^T ds\Bigg) \\ &\quad \Big(f_i\big(\tilde{x}(\tau_i(t),t,x(t),\bar{\tau}(t))\big) - f_{i+1}\big(\tilde{x}(\tau_i(t),t,x(t),\bar{\tau}(t))\big)\Big). \qquad (135) \end{aligned}$$

$\frac{d\psi_{i,t}}{d\xi}(t)$ is obtained in a backwards recursive manner as follows:

$$\begin{aligned}\frac{d\psi_{i,t}}{d\xi}(t) &= \Phi_{i,t,x(t)}(\tau_i(t),\tau_{i+1}(t))\frac{d\psi_{i+1}}{d\xi}(t))\\ &\quad+\int_{\tau_i(t)}^{\tau_{i+1}(t)}\Phi_{i,t,x(t)}(\tau_{i+1}(t),s)\left(\frac{\partial^2 L}{\partial t\partial x}\big(\tilde{x}(s,t,x(t),\bar{\tau}(t)),s,t\big)\right)^T ds,\end{aligned} \tag{136}$$

for $i = N-1, N-2, ..., 1$, and

$$\frac{d\psi_{N,t}}{d\xi}(t) = \int_{\tau_N(t)}^{T}\Phi_{i,t,x(t)}(\tau_{N+1}(t),s)\left(\frac{\partial^2 L}{\partial t\partial x}\big(\tilde{x}(s,t,x(t),\bar{\tau}(t)),s,t\big)\right)^T ds. \tag{137}$$

Proof 5.5 *Again, from Chapter 2, equation (13), we have*

$$\frac{\partial J}{\partial \tau_i}(t,x,\tau) \;=\; p(\tau_i,t,x,\bar{\tau})^T\Big(f_i\big(\tilde{x}(\tau_i,t,x,\bar{\tau})\big)-f_{i+1}\big(\tilde{x}(\tau_i,t,x,\bar{\tau})\big)\Big),$$

Plug the above equation into equation (133), we have

$$\begin{aligned}&\frac{\partial J}{\partial \tau_i}(t,x,\tau)\\ =\;&\Big(\Phi_{i,t,x}(\tau_i,\tau_{i+1})p(\tau_{i+1},t,x,\tau)+\int_{\tau_i}^{\tau_{i+1}}\Phi_{i,t,x}(\tau,s)\frac{\partial L}{\partial x}\big(\tilde{x}(s,t,x,\bar{\tau}),s,t\big)^T ds\Big)^T\\ &\Big(f_i\big(\tilde{x}(\tau_i,t,x,\bar{\tau})\big)-f_{i+1}\big(\tilde{x}(\tau_i,t,x,\bar{\tau})\big)\Big)\end{aligned} \tag{138}$$

Similar to the case for single switching, use Lemma 5.2 and (138) with two specific values: $(t,x(t),\bar{\tau}(t))$, and $(t+\Delta t, x(t+\Delta t),\bar{\tau}(t))$, to obtain $\frac{d\phi_{i,t}}{d\xi}(t)$:

$$\begin{aligned}\frac{d\phi_{i,t}}{d\xi}(t) &= \lim_{\Delta t\to 0}\frac{1}{\Delta t}\left(\frac{\partial J}{\partial \tau_i}(t+\Delta t,x(t+\Delta t),\bar{\tau}(t))-\frac{\partial J}{\partial \tau_i}(t,x(t),\bar{\tau}(t))\right)\\ &= \Bigg(\left(\lim_{\Delta t\to 0}\frac{1}{\Delta t}p(\tau_{i+1},t+\Delta t,x(t+\Delta t),\bar{\tau}(t))-p(\tau_{i+1},t,x(t),\bar{\tau}(t))\right)^T\\ &\quad\Big(\Phi_{i,t,x(t)}(\tau_i(t),\tau_{i+1}(t))\Big)^T\\ &\quad+\int_{\tau_i(t)}^{\tau_{i+1}(t)}\frac{\partial^2 L}{\partial t\partial x}\big(\tilde{x}(s,t,x(t),\bar{\tau}(t)),s,t\big)\left(\Phi_{i,t,x(t)}(\tau_{i+1}(t),s)\right)^T ds\Bigg)\\ &\quad\Big(f_i\big(\tilde{x}(\tau_i(t),t,x(t),\bar{\tau}(t))\big)-f_{i+1}\big(\tilde{x}(\tau_i(t),t,x(t),\bar{\tau}(t))\big)\Big).\end{aligned}$$

After some simplification

$$
\begin{aligned}
\frac{d\phi_{i,t}}{d\xi}(t) &= \Bigg(\left(\frac{d\psi_{i+1,t}}{d\xi}(t) \right)^T \left(\Phi_{i,t,x(t)}(\tau_i(t), \tau_{i+1}(t)) \right)^T \\
&\quad + \int_{\tau_i(t)}^{\tau_{i+1}(t)} \frac{\partial^2 L}{\partial t \partial x} \big(\tilde{x}(s,t,x(t),\bar{\tau}(t)), s, t \big) \left(\Phi_{i,t,x(t)}(\tau_{i+1}(t), s) \right)^T ds \Bigg) \\
&\quad \Big(f_i\big(\tilde{x}(\tau_i(t), t, x(t), \bar{\tau}(t))\big) - f_{i+1}\big(\tilde{x}(\tau_i(t), t, x(t), \bar{\tau}(t))\big) \Big).
\end{aligned} \tag{139}
$$

Notice that the above equation used the definition of $\psi_{i,t}(\xi)$ defined in (134). Furthermore, plug $(t, x(t), \bar{\tau}(t))$ and $(t + \Delta t, x(t + \Delta t), \bar{\tau}(t))$ into (133), the derivative $\frac{d\psi_{i,t}}{d\xi}(t)$ can be obtained

$$
\begin{aligned}
\frac{d\psi_{i,t}}{d\xi}(t) &= \Phi_{i,t,x(t)}(\tau_i(t), \tau_{i+1}(t)) \frac{d\psi_{i+1}}{d\xi}(t)) \\
&\quad + \int_{\tau_i(t)}^{\tau_{i+1}(t)} \Phi_{i,t,x(t)}(\tau_{i+1}(t), s) \left(\frac{\partial^2 L}{\partial t \partial x} \big(\tilde{x}(s,t,x(t),\bar{\tau}(t)), s, t \big) \right)^T ds.
\end{aligned}
$$

Observe that since $p(\tau_{i+1}(t), t, x(t), \bar{\tau}(t) = 0$, then

$$
\frac{d\psi_{N,t}}{d\xi}(t) = \int_{\tau_N(t)}^{T} \Phi_{i,t,x(t)}(\tau_{N+1}(t), s) \left(\frac{\partial^2 L}{\partial t \partial x} \big(\tilde{x}(s,t,x(t),\bar{\tau}(t)), s, t \big) \right)^T ds.
$$

Thus the proof is completed.

Using Theorem 5.3 and (133), the switching evolution $\dot{\bar{\tau}}(t)$ is obtained for $t \in [\tau_0, \tau_1(t))$. When $t = \tau_1(t)$, we "freeze" $\tau_1(t)$, switch the mode from q_1 to q_2, set $\tau_0 = \tau_1(t)$, $\bar{\tau}(t) = [\tau_2(t), \tau_3(t), ..., \tau_N(t)]^T$, $N = N - 1$ and restart the continuous process. This process is repeated until all switchings are triggered and the entire evolution is obtained.

5.2 Numerical Algorithms

Similar to the observer-based real-time optimal control, we now propose numerical algorithms that are based on the continuous process. A few remarks should be made. First, note that, as an discrete algorithm, it is no longer possible to track the cost function for all time. Rather, one can only hope to update the switching times so that they are optimal with respect to the cost function sampled at some frequency. Hence, the continuous update of the cost function L is now a discrete update with a sampling period ΔT, which is assumed to be much larger than the time-step Δt to ensure convergence of the switching times. Second, we can use the gradient formula (120) and (128) developed in the continuous process in the discrete algorithms. Third, the algorithms are similar in spirit to the algorithms proposed in Chapter 2, and their convergence analysis are similar. The difference is that for this class of problems, we can use the explicit gradient formula derived. This gradient formula is expressed explicitly in terms of the rate of change of the cost function.

At time t, we use the notation t_s to denote the time when the cost function is last updated. Note that $t - t_s \leq \Delta T$. We denote $q(t)$ to be current mode of operation at time t. Then, the control variable at time t is denoted as $\bar{\tau}(t) = [\tau_{q(t)}(t), \tau_{q(t)+1}(t), \ldots, \tau_N(t)]^T$. Note that this switching time vector is merely the control variable at time t, and it may or may not be optimal with respect to the cost-to-go (as opposed to the case for the continuous process, $\bar{\tau}(t)$ was formulated to be always optimal). Finally, we define the the time-varying feasible set: $\Lambda(t) = \{\bar{\tau}(t) : t = \tau_{q(t)}(t) \leq \tau_{q(t)+1}(t) \leq \ldots, \leq \tau_N(t) \leq \tau_{N+1}(t) = T\}$.

We first propose the Newton-Raphson like algorithm.

Algorithm 5.1 *Assume that f_i and L are three-times continuously differentiable. At time $t \in [t_s, t_s + \Delta T])$, the cost-to-go performance measure is $J(t, x(t), \bar{\tau}(t)) = \int_t^T L(\tilde{x}(s, t_s, x(t), \bar{\tau}(t)), s, t_s)ds$.*

Step 1: Compute $\tilde{h}(t)$, defined as the projection of $-\frac{\partial^2 J}{\partial \bar{\tau}^2}\big(t, x(t), \bar{\tau}(t)\big)^{-1} \frac{\partial J}{\partial \bar{\tau}}(t+\Delta t, x(t+\Delta t), \bar{\tau}(t))$ onto the feasible set $\Lambda(t)$ at $\bar{\tau}(t)$. $\frac{\partial J}{\partial \bar{\tau}}(t + \Delta t, x(t + \Delta t), \bar{\tau}(t)) = \frac{d\phi_t}{d\xi}(t)\Delta t + O(\Delta t^2)$ is computed by equation (135) using Theorem 5.3.

Step 2: Compute $\gamma(t) := \max\{\gamma \leq 1 \mid \bar{\tau}(t) + \gamma\tilde{h}(t) \in \Lambda(t)\}$.

Step 3. Define $h(t) := \gamma(t)\tilde{h}(t)$, and set

$$\bar{\tau}(t + \Delta t) \;:=\; \bar{\tau}(t) + h(t). \tag{140}$$

When $t > \tau_{q(t)}$, the switching transition is triggered and the mode of the system is switched from $q(t)$ to $q(t)+1$. $\bar{\tau}(t)$ is set to $[\tau_{q(t)+1}(t), ..., \tau_N(t)]$. From this point on, the switching time $\tau_{q(t)}$ is considered frozen, since the switching has already occurred. The algorithm stops when all switchings have occurred, otherwise, set $t = t + \Delta t$. If $t \geq t_s + \Delta T$, set $t_s = t_s + \Delta T$ and update $L(\tilde{x}, s, t_s)$. Go to Step 1.

At time t, this algorithm updates the switching times $\bar{\tau}(t)$ towards a local optimal solution for the cost-to-go performance measure $J = \int_t^T L(\tilde{x}(s, t_s, x(t), \bar{\tau}(t)), s, t_s)ds$. Note that this local optimal solution is function of the current sample (t_s) of the time-varying instantaneous cost function L, and we denote it by $\bar{\tau}^\star(t_s)$. Due to Bellman's Optimality Principle, $\bar{\tau}^\star(t_s)$ is constant during the interval $[t_s, t_s+\Delta T)$. Hence, $\bar{\tau}^\star(t_s)$ satisfies the necessary optimality conditions

$$\frac{\partial J}{\partial \bar{\tau}}(t, x(t), \bar{\tau}^\star(t_s)) = 0, \tag{141}$$

and $\frac{\partial^2 J}{\partial \bar{\tau}^2}(t, x(t), \bar{\tau}^\star(t_s))$ is positive definite. Now, we present the convergence analysis of this algorithm, which follows directly from the proof of Theorem 3.1, the difference is that this convergence analysis is stated with respect to $\bar{\tau}^\star(t_s)$. It should be noted that, since each execution of Algorithm 5.1 obtains the same Newton step as Algorithm 3.1, the convergence analysis of Algorithm 5.1 is similar to Algorithm 3.1. However, note that in this algorithm, we perform the Newton step with the explicit gradient formula.

The convergence analysis is again presented in the form of convergence rate instead of in an asymptotic sense.

Theorem 5.4 *At time $t \in [t_s, t_s + \Delta T)$, suppose that $\bar{\tau}^\star(t_s)$ satisfies the necessary optimality condition (100) lying in the interior of the feasible set $\Lambda(t)$, and that $H(t) = \frac{\partial^2 J}{\partial \bar{\tau}^2}\big(t, x(t), \bar{\tau}(t)\big)$ is positive definite. Then there exist constants $\delta > 0$ and $K > 0$ such that, if $||\bar{\tau}(t) - \bar{\tau}^\star(t_s)|| < \delta$; $\Delta t < \delta$; $\gamma(t) = 1$; and $\tau_{q(t)} + \Delta t > t$, then*

$$||\bar{\tau}(t + \Delta t) - \bar{\tau}^\star(t_s)|| \leq \ K\Big(||\bar{\tau}(t) - \bar{\tau}^\star(t_s)||^2 + \Delta t||\bar{\tau}(t) - \bar{\tau}^\star(t_s)||\Big). \tag{142}$$

Proof 5.6 *Follows from Theorem 3.1, by setting* $e(t) = 0$.

The convergence analysis shows that the speed of Algorithm 5.1 is close to quadratic, if Δt is small. Furthermore, to have good performance for the algorithm, Δt needs to be much smaller than ΔT to ensure enough Newton steps are taken.

There are two drawbacks for this algorithm. First, like all Newton-like algorithms, if current iterate of $\bar{\tau}(t)$ is not close to the optimal solution $\bar{\tau}^\star(t_s)$, then the algorithm may not converge or even progress towards $\bar{\tau}^\star(t_s)$. Second, the Hessian $H(t)$ may not be positive definite. These two drawbacks can be avoided in the slower converging gradient descent-based algorithm, which is presented below.

Algorithm 5.2 *Assume that f_i and L are three-times continuously differentiable. At time $t \in [t_s, t_s + \Delta T])$, the cost-to-go performance measure is $J(t, x(t), \bar{\tau}(t)) = \int_t^T L(\tilde{x}(s, t_s, x(t), \bar{\tau}(t)), s, t_s)ds$.*
Step 1: Compute $\tilde{h}(t)$, defined as the projection of $\frac{\partial J}{\partial \bar{\tau}}(t+\Delta t, x(t+\Delta t), \bar{\tau}(t))$ onto the feasible set $\Lambda(t)$ at $\bar{\tau}(t)$. $\frac{\partial J}{\partial \bar{\tau}}(t + \Delta t, x(t + \Delta t), \bar{\tau}(t)) = \frac{d\phi_t}{d\xi}(t)\Delta t + O(\Delta t^2)$ is computed by equation (135) using Theorem 5.3.
Step 2: Compute $\gamma(t) := \max\{\gamma \leq 1 \mid \bar{\tau}(t) + \gamma\tilde{h}(t) \in \Lambda(t)\}$.
Step 3. Define $h(t) := \gamma(t)\tilde{h}(t)$, and set

$$\bar{\tau}(t + \Delta t) \;:=\; \bar{\tau}(t) + h(t). \tag{143}$$

When $t > \tau_{q(t)}$, the switching transition is triggered and the mode of the system is switched from $q(t)$ to $q(t) + 1$. $\bar{\tau}(t)$ is set to $[\tau_{q(t)+1}(t), ..., \tau_N(t)]$. From this point on, the switching time $\tau_{q(t)}$ is considered frozen, since the switching has already occurred. The algorithm stops when all switchings have occurred, otherwise, set $t = t + \Delta t$. If $t \geq t_s + \Delta T$, set $t_s = t_s + \Delta T$ and update $L(\tilde{x}, s, t_s)$. Go to Step 1.

This algorithm utilizes gradient decent direction as opposed to Newton-Raphson decent direction. Such algorithms are known to converge globally to Kuhn-Tucker stationary points. They also have linear convergence rate, which is shown by the following convergence rate analysis.

Theorem 5.5 *At time t, suppose that $\bar{\tau}^\star(t_s)$ is a stationary point lying in the interior of the feasible set $\Lambda(t)$, which is the target of our algorithm. There exist constant $\delta > 0$, and $K_1 \in (0,1)$ such that, if $||\bar{\tau}(t) - \bar{\tau}^\star(t_s)|| < \delta$; $\Delta t < \delta$ and $\tau_{q(t)} + \Delta t > t$; then*

$$||\bar{\tau}(t+\Delta t) - \bar{\tau}^\star(t_s)|| \leq K_1 ||\bar{\tau}(t) - \bar{\tau}^\star(t_s)|| \tag{144}$$

Proof 5.7 *Follows from Theorem 3.2, by setting $e(t) = 0$.*

A few conclusions can be drawn from the above convergence analysis. First, $\gamma(t)$ determines the constant K_1, which dictates the speed of convergence. This algorithm converges faster when K_1 becomes closer to 0. Furthermore, without the calculations of the Hessian, this algorithm is computationally much faster than the Algorithm 5.1. Hence, the gradient descent algorithm complements the Newton-Raphson algorithm since it addresses the drawbacks of using the Newton update direction as mentioned previously, with the trade-off of slower convergence rate. Therefore, one can use a hybrid algorithm that switches from the gradient descent algorithm to the Newton-Raphson algorithm when the gradient $\frac{\partial J}{\partial \bar{\tau}}(t, x(t), \bar{\tau}(t))$ is smaller than a pre-defined threshold.

5.3 Adaptive Double-tank Example

In this section we show the effect of the results presented in this chapter through a simple but non-linear problem. We consider the problem of controlling the fluid levels in a double-tank system, as shown in Figure 20. This problem was initial studied as a switching-time optimization example in [8], but we re-frame this problem in a different setting.

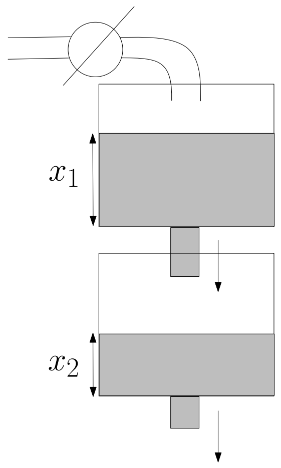


Figure 20: The double-tank process

The goal of this numerical example is to control the fluid level of the second tank so that it minimizes the integral over the distance between the trajectory to a reference point. Furthermore, instead of assuming that the reference point is static and given a priori (as in the case of [8]), we assume that the desired fluid level of the second tank (the reference point) is provided and updated during the operation of the system. Hence the reference point evolves continuously with an unknown dynamics. As seen in Figure 20, fluid flows from a valve to the first tank, and then from the first tank to the second tank. For this example, fluid in the second tank is always discharging. Similar to many chemical processes, the valve can only be open or closed. As such, this system is inherently a switched system.

We denote the fluid level of the first tank at time t by $x_1(t)$ and the second tank by $x_2(t)$. The state of this system is denoted by $x(t) = [x_1(t), x_2(t)]^T$. Based on Torricelli's Principle, the fluid in the tank is discharging at a rate proportional to the square root of the current fluid level. Hence the dynamics of this switched system can be described by the following equations:

$$\dot{x}(t) = f_1(x(t)) = \begin{bmatrix} -\alpha_1\sqrt{x_1(t)} \\ \alpha_1\sqrt{x_1(t)} - \alpha_2\sqrt{x_2(t)} \end{bmatrix}, \tag{145}$$

or

$$\dot{x}(t) = f_2(x(t)) = \begin{bmatrix} -\alpha_1\sqrt{x_1(t)} + u \\ \alpha_1\sqrt{x_1(t)} - \alpha_2\sqrt{x_2(t)} \end{bmatrix}, \tag{146}$$

where f_1 corresponds to the mode (mode 1) when the valve is closed, and f_2 corresponds to the mode (mode 2) when the valve is open. In (145) and (146), α_1 and α_2 denotes the flow rate of the tank, and u denotes the inflow rate. For this example, we assume that we start with mode 2 and switch once during the execution of the system to mode 1. $\tau(t)$ is the optimal switching time at time t and $r(t)$ is the reference point at time t.

Since the objective of this problem is to control the average water level of the second tank to a reference point only available on-line, we define the time-varying instantaneous cost function as $L(t, \tilde{x}(s,t,x(t),\tau(t)), s) = \frac{1}{2}(\tilde{x}_2(s,t,x(t),\tau(t)) - r(t))^2$. Hence the cost-to-go functional is as follows:

$$J(t, x(t), \tau(t)) = \frac{1}{2}\int_t^T (\tilde{x}_2(s,t,x(t),\tau(t)) - r(t))^2 ds. \tag{147}$$

For this example, since $\frac{\partial^2 L}{\partial t \partial x} = [0, -\dot{r}(t)]$, the evolution of the optimal switching time can be simplified using equation (120) as:

$$\begin{aligned} \dot{\tau}(t) \quad = \quad & -\Big(\frac{\partial^2 J}{\partial \tau^2}(t, x(t), \tau(t))\Big)^{-1}[0, -\dot{r}(t)]\int_{\tau(t)}^T \Big(\Phi_{t,x(t)}(\tau(t), s)\Big)^T ds \\ & \Big(f_1(\tilde{x}(\tau(t), t, x(t), \tau(t))) - f_2(\tilde{x}(\tau(t), t, x(t), \tau(t)))\Big). \end{aligned} \tag{148}$$

As we can see from (148), in order to update the switching time so that it is always optimal with respect to the cost-to-go (147), we need to obtain the derivative of the reference point. This information is the rate of change of the cost function for this particular problem. Furthermore, note that if the rate of change of the cost function is 0, then the switching time should not be updated ($\dot{\tau}(t) = 0$), since it is already optimal and it should remain optimal because of the Bellman's Principle.

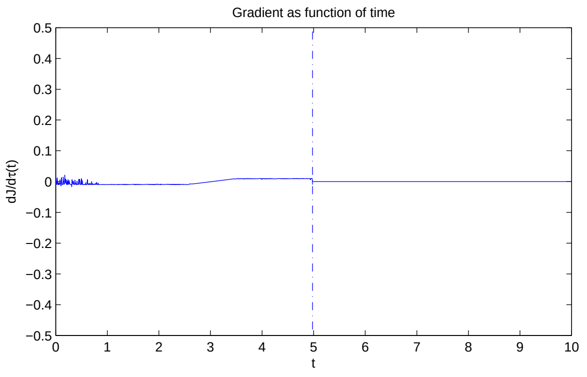


Figure 21: Plot of the gradient $\frac{dJ}{d\tau}(t, x(t), \tau(t))$ as a function of time t. The vertical dotted line corresponds to the time when mode switch occurred. After the mode switch, the gradient is set to 0.

In this example, we set $\alpha_1 = \alpha_2 = 1$ and $u = 2$. The time horizon T is set to 10. To implement the continuous process, we use the discrete algorithm (Algorithm 5.1) and sample the reference point $r(t)$ with step-size $\Delta T = 0.05$. For the time-step of the algorithm, we use a smaller step-size ($\Delta t = 0.005$) to update the switching time and the system. The reference point evolution supplied for the purpose of evaluation is a parabola $r(t) = -\frac{1}{9}(t-3)^2 + 1.7$.

Using the method outlined in this chapter, the resultant gradient $\frac{dJ}{d\tau}(t, x(t), \tau(t))$ for the cost-to-go (147) is shown in the Figure 21. The gradient is close to 0 for all time which is the expected and desired result. The final state trajectories $x_1(t)$ and $x_2(t)$ of the system and the continuous varying reference signal $r(t)$ as a function of time is shown in Figure 22. For this example, the reference signal started at $r(0) = 0.7$. When the switching takes place at time $t = \tau(t) = 4.8$, the final reference point is 1.3. As a result, the final state trajectory $x_2(t)$ minimizes the the square tracking error with respect to the final reference point $r(4.8)$, which is 1.3.

Starting at optimal initial condition $\tau(0) = 1.86$ (corresponds to $r(0) = 0.7$), the evolution of the optimal switching time using (148) is shown in Figure 23. The diagonal line represents $\tau(t) = t$ and mode switch takes place when $\tau(t)$ hits this line.

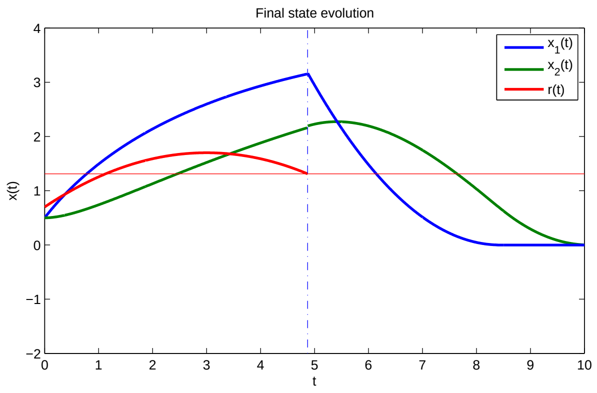


Figure 22: Plot of the final state trajectory and the reference point evolution of the system. The vertical dotted line corresponds to the time when the mode switch occurred and the horizon line is the final desired reference point 1.3.

At this point, the system switches from mode 2 to mode 1 and the system is no longer controlled.

5.4 Conclusions

In this chapter we address the problem of adapting the optimal switching times of a switched system in an on-line environment where the instantaneous cost function is assumed to be continuously varying. We proposed an approach to obtain the trajectory of the optimal switching times with respect to the on-line cost-to-go function. We found that to maintain this optimal switching time evolution, on-line information in the form of *rate of change* of the instantaneous cost is required. We proposed a continuous process and numerical algorithms to update the switching times so that they remain optimal with respect to the time-varying cost function.

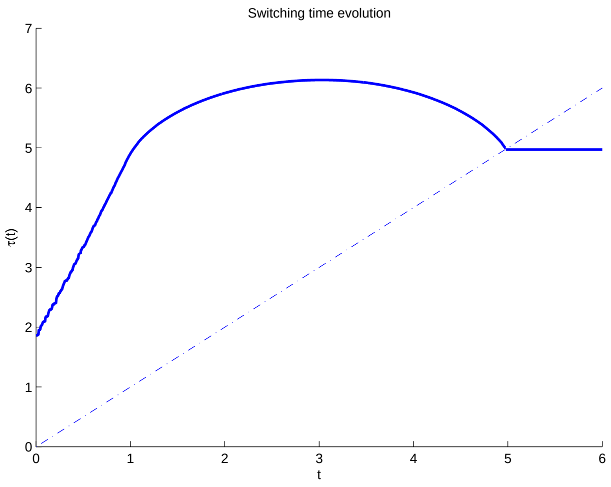


Figure 23: Plot of the switching time evolution $\tau(t)$ as function of time. The diagonal line is the line $\tau(t) = t$ and mode switch takes place if $\tau(t)$ hits this line. The switching time does not change after this point since the mode switch has already occurred.

CHAPTER VI

RECEDING HORIZON OPTIMAL TIMING CONTROL

Optimal control of switched systems requires frequent computations of the state trajectory as a function of the control variables. However, when the switched system is required to operate for a long time, it may be no longer feasible to compute the optimal solution due to the length of the simulated state trajectory. Motivated by this idea, in this chapter we propose a receding horizon techniques to provide sub-optimal control laws for the switched system by using a much smaller time horizon.

Given an autonomous switched system

$$\dot{x} = f_i(x), t \in [\tau_{i-1}, \tau_i), i = 1, 2, ..., N+1, \tag{149}$$

where $x(t) \in \mathbb{R}^n$ is the state variable, initial condition $x(0) = x_0$ and final time T are given. Modal functions $f_i : \mathbb{R}^n \to \mathbb{R}^n$ correspond to a fixed mode sequence $\bar{q} = \{1, 2, ..., N+1\}$ and the notations $\tau_0 = 0$, $\tau_{N+1} = T$ are used. The control input of this system is the switching time vector $\bar{\tau} = [\tau_1, \tau_2, ..., \tau_N]^T$, where it must satisfy the constraint $\bar{\tau} \in \Lambda(t)$ and

$$\Lambda := \{\bar{\tau} = [\tau_1, ..., \tau_N]^T : 0 = \tau_0 \le \tau_1 \le ... \le \tau_N \le \tau_{N+1} = T\}. \tag{150}$$

We again consider the timing optimization problem

$$\min_{\bar{\tau} \in \Lambda} J = \int_0^T L(x(t))dt, \tag{151}$$

with a given trajectory cost function $L : \mathbb{R}^n \to \mathbb{R}$.

There are many situations where one may desire not to use a fixed time horizon T. In many applications, it may not be feasible to compute the optimal switching times for a long time horizon. The terminal time may not be known exactly, or it can be changed during operation of the plant.

Therefore, we adopt an approach in which a smaller sliding window is selected, and the switching times are optimized over this window. The goal is to preserve the optimality of the switching times as the window slides. In other words, we wish to provide a method to update the switching times so that they are always optimal with respect to the sliding window. Due to the real time constraint, the computation for this algorithm must be computationally feasible.

Similar to the previous chapters of this thesis, we first propose a continuous process that defines a differential equation for the evolution of the optimal switching time vector when the the sliding window progress in real-time. Then we turn the continuous process into a discrete algorithm.

6.1 Continuous Process

To address this problem, first note that the cost functional that we wish to optimize over is no longer fixed. Instead, we use a cost-to-go functional that is defined by the sliding window. Let T_s denotes the length of the sliding window, the cost-to-go functional is defined as :

$$J(t, x(t), \bar{\tau}) = \int_{t}^{t+T_s} L(\tilde{x}(s))ds. \tag{152}$$

The instantaneous cost L is evaluated over the future state trajectory, denoted by $\tilde{x}(s)$, which is defined as follows:

$$\dot{\tilde{x}}(s) \ = \ F(\tilde{x}, s), s \in [t, t + T_s], \tag{153}$$

with the initial condition $\tilde{x}(t) = x(t)$ and given $\bar{\tau}$, where $F(x, t)$ is defined as $F(x, t) \ :=$ $f_i(x)$ for all $t \in [\tau_{i-1}, \tau_i), \ i = 1, \ldots, N + 1$.

The current and future trajectory is illustrated in Figure 24, where the solid curve represents the past trajectory and the dotted curve represents the future (or projected) trajectory.

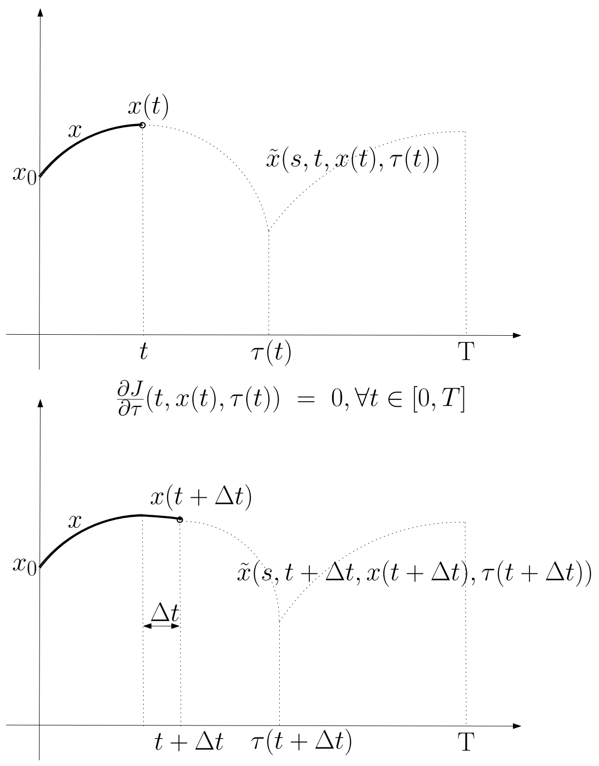

$x(t)$
x
x_0
$\tilde{x}(s,t,x(t),\tau(t))$
t
$\tau(t)$
T
$\frac{\partial J}{\partial \tau}(t,x(t),\tau(t)) = 0, \forall t \in [0,T]$
$x(t+\Delta t)$
x
x_0
$\tilde{x}(s,t+\Delta t,x(t+\Delta t),\tau(t+\Delta t))$
Δt
$t+\Delta t$
$\tau(t+\Delta t)$
T

Figure 24: Current state and the state simulated into the future.

In this section we denote the problem of optimizing the cost-to-go functional (152) at time t by Π_t. The approach of this chapter allows the means to compute the trajectory of the switching time vector, such that it is an approximation of a local optimal solution for the cost-to-go functional at each time instant t. This local optimal switching time vector is a function of time t as well, and is defined as:

$$\bar{\tau}(t) = \min_{\bar{\tau}} J(t, x(t), \bar{\tau}), t \geq t_0, \tag{154}$$

for an initial time t_0. To compute trajectory $\bar{\tau}(t)$, we develop an iterative process which is characterized by a differential equation. By following this differential equation, the switching times are updated so that they remain locally optimal to the sliding window at each time instant t. In the continuous problem perspective, the goal is to establish the time derivative of the optimal switching time vector $\bar{\tau}(t)$ at time t, namely $\dot{\bar{\tau}}(t)$, so that optimality is conserved with respect to the sliding window. In other words, we aim to compute $\dot{\bar{\tau}}(t)$, so that if $\bar{\tau}(t)$ is a solution point for

Π_t, then $\bar{\tau}(t+dt)$ computed by this continuous process

$$\bar{\tau}(t+dt) = \bar{\tau}(t) + \dot{\bar{\tau}}(t)dt \tag{155}$$

is a solution point for Π_{t+dt}, and $\frac{\partial J}{\partial \bar{\tau}}(t+dt, x(t+dt), \bar{\tau}(t+dt)) = 0$.

Note that if we obtain the optimal switching time vector $\bar{\tau}(t)$ and substitute it in (152), then $J(t, x(t), \bar{\tau}(t))$ is locally optimal and

$$\frac{\partial J}{\partial \bar{\tau}}(t, x(t), \bar{\tau}(t)) = 0, \tag{156}$$

Since we wish to preserve optimality, it follows that the optimal switching time trajectory $\bar{\tau}(t)$ also satisfies:

$$\frac{d}{dt}\Big(\frac{\partial J}{\partial \tau}(t, x(t), \bar{\tau}(t))\Big) = 0. \tag{157}$$

With (157), we present the following theorem.

Theorem 6.1 *Given $\bar{\tau}(t)$ as a local minimum for Π_t. The second derivative of J with respect to $\bar{\tau}$ is strictly positive. Then the time derivative of $\bar{\tau}(t)$ can be determined by:*

$$\dot{\bar{\tau}}(t) = -\left(\frac{\partial^2 J}{\partial \bar{\tau}^2}(t, x(t), \bar{\tau}(t))\right)^{-1} \frac{\partial^2 J}{\partial t \partial \bar{\tau}}(t, x(t), \bar{\tau}(t)) + \frac{\partial^2 J}{\partial x \partial \bar{\tau}}(t, x(t), \bar{\tau}(t))\dot{x}(t). \tag{158}$$

Proof 6.1 *Directly from (157),*

$$\begin{aligned} & \frac{d}{dt}\Big(\frac{\partial J}{\partial \tau}(t, x(t), \bar{\tau}(t))\Big) \\ = & \frac{\partial^2 J}{\partial t \partial \bar{\tau}}(t, x(t), \bar{\tau}(t)) + \frac{\partial^2 J}{\partial x \partial \bar{\tau}}(t, x(t), \bar{\tau}(t))\dot{x}(t) + \frac{\partial^2 J}{\partial \bar{\tau}^2}(t, x(t), \bar{\tau}(t))\dot{\bar{\tau}}(t) = 0. \end{aligned} \tag{159}$$

Rearranging the terms in (159) brings (158).

Furthermore, as a result of Taylor expansion, we can obtain an alternative equation for the terms $\frac{\partial^2 J}{\partial t \partial \bar{\tau}}(t, x(t), \bar{\tau}(t)) + \frac{\partial^2 J}{\partial x \partial \bar{\tau}}(t, x(t), \bar{\tau}(t))\dot{x}(t)$ as:

$$\frac{\partial^2 J}{\partial t \partial \bar{\tau}}(t, x(t), \bar{\tau}(t)) + \frac{\partial^2 J}{\partial x \partial \bar{\tau}}(t, x(t), \bar{\tau}(t))\dot{x}(t) = \lim_{dt \to 0} \frac{1}{dt}\frac{\partial J}{\partial \bar{\tau}}(t+dt, x(t+dt), \bar{\tau}(t)). \tag{160}$$

Plug (160) in (155), we obtain the equation to compute $\bar{\tau}(t+dt)$ based on $\bar{\tau}(t)$.

$$\bar{\tau}(t+dt) = \bar{\tau}(t) - \lim_{dt\to 0}\left(\frac{\partial^2 J}{\partial\bar{\tau}^2}(t, x(t), \bar{\tau}(t))\right)^{-1}\frac{\partial J}{\partial\tau}(t+dt, x(t+dt), \bar{\tau}(t)). \qquad (161)$$

Note that equation (161) defines a continuous process, which can be turn into a discrete algorithm. However, due the fact that the continuous process is turned into a discrete algorithm by using a finite time-step Δt instead of dt, the resultant algorithm only produces sub-optimal switching time evolutions. This is detailed in the next section.

6.2 Numerical Algorithm

In this section, we extend the approach from a continuous process to an implemented algorithm. The real-time algorithm is divided into two phases. The first phase of the algorithm is to re-compute the optimal switching times so that $\bar{\tau}(t_0)$ is a local minimum for the problem Π_{t_0}, for a given time horizon T. This is needed since the continuous process requires a locally optimal starting point $\bar{\tau}(t_0) = \min_{\bar{\tau}} J(t_0, x(t_0), \bar{\tau})$. Furthermore, this computation must be performed on-line, whenever the algorithm is required to be restarted.

The time horizon T is selected so that it is large enough to enable the convergence of the on-line algorithm. Once it converges, we set the current time to be t_0, as well as picking a sliding window length $T_s = T - t_0$, and the second phase of the algorithm starts. The second phase updates $\bar{\tau}(t)$ so that it becomes an estimate of local minimum for the next time-step.

The first phase of the algorithm uses the on-line algorithm proposed in the Chapter 3 of this thesis. For self-containment of this chapter, we re-iterate the algorithm here. First define the Hessian

$$H(t) = \frac{\partial^2 J}{\partial\bar{\tau}^2}(t, \tilde{x}(t), \bar{\tau}(t)). \qquad (162)$$

Denote the mode at time t as $q(t)$, then the feasible set can be defined as: $\Lambda(t) =$

$\{\bar{\tau}(t) : t = \tau_{q(t)-1}(t) \leq \tau_{q(t)}(t) \leq \ldots, \leq \tau_N(t) \leq \tau_{N+1}(t) = T\}$. The first-phase of the algorithm is presented below.

Algorithm 6.1 (First Phase) *At time t, do the following steps:*

Step 1. Compute $\tilde{h}(t)$, defined as the projection of $-H(t))^{-1}\frac{\partial J}{\partial \tau}(t+\Delta t, \tilde{x}(t+\Delta t), \bar{\tau}(t))$ onto the feasible set $\Lambda(t)$.

Step 2. Compute $\gamma(t) := \max\{c \leq 1 \mid \bar{\tau}(t) + c\tilde{h}(t) \in \Lambda(\mathrm{t})\}$.

Step 3. Define $h(t) := \gamma(t)\tilde{h}(t)$, and set $\bar{\tau}(t + \Delta t) \ = \ \bar{\tau}(t) + h(t)$.

The gradient $\frac{\partial J}{\partial \tau}(t + \Delta t, \tilde{x}(t + \Delta t), \bar{\tau}(t))$ is computed as follows. First the projected state trajectory $\tilde{x}(s)$ is computed by (153) over the time interval $s \in [t+\Delta t, T]$ with the switching time vector $\bar{\tau}(t)$. Then a projected costate trajectory is computed backward using the costate equation introduced in (9) with the initial condition $p(\tau_{N+1}) = 0$. The gradient is then computed by (13).

The above algorithm is executed at each time-step until the projection of the gradient $\frac{\partial J}{\partial \tau}(t + \Delta t, \tilde{x}(t + \Delta t), \bar{\tau}(t))$ onto the feasible set $\Lambda(t)$ is below a predefined bound, at which point we obtain a local minimum for Π_{t_0}. This concludes the first phase of the iterative process.

Now, we extend the continuous process described in the previous section into a discrete algorithm, by changing the infinitesimal time interval dt to be a finite interval Δt. The resultant discrete algorithm is an estimate of the continuous process (155). The second phase of the algorithm is presented below.

Algorithm 6.2 (Second Phase) *At time t, do the following steps:*

Step 1. Compute $\tilde{h}(t)$, defined as the projection of $-H(t)^{-1}\frac{\partial J}{\partial \tau}(t+\Delta t, \tilde{x}(t+\Delta t), \bar{\tau}(t))$ onto the feasible set $\Lambda(t)$.

Note that the gradient is computed using the same equation as (13), however the state and costate trajectories are computed over the time interval $[t + \Delta t, t + \Delta t + T_s]$.

Step 2. Compute $\gamma(t) := \max\{\gamma \leq 1 \mid \bar{\tau}(t) + \gamma\tilde{h}(t) \in \Lambda(\mathrm{t})\}$.

Step 3. Define $h(t) := \gamma(t)\tilde{h}(t)$, *and set* $\bar{\tau}(t+\Delta t) \;=\; \bar{\tau}(t) + h(t)$.

The difference between Algorithm 6.1 and Algorithm 6.2 is the time horizon used to compute the gradients and the Hessians ($[t+\Delta t, T]$ versus $[t+\Delta t, t+\Delta t+T_s]$). The combined algorithm produces an approximation of the continuous process (155), hence the switching time evolution $\bar{\tau}(t)$ is not always optimal, but only considered to be sub-optimal. This algorithm will be used in the next chapter, where we present a realistic application that requires a receding-horizon optimal control framework.

6.3 Conclusions

In this chapter we present a receding-horizon algorithm for optimal timing control of switched system. We propose to define the time horizon of the optimal control problem to be a sliding window, and we obtain a continuous process that produces an evolution of optimal switching times that is always optimal as the window slides. Then we present a numerical algorithm that approximate this process, resulting in a sub-optimal switching times. The results of this chapter will be used for an application that requires a receding-horizon optimal control framework.

CHAPTER VII

APPLICATION OF ON-LINE OPTIMAL CONTROL OF SWITCHED SYSTEMS: PILOT DECISION SUPPORT FRAMEWORK

In this chapter we aim to demonstrate the use of the real-time algorithms proposed by this thesis through an application of optimal formation and coverage control of a networked system. This application is demonstrated through a realistic simulation framework consisting of a number of Unmanned Aerial Vehicles (UAVs) that interact in a virtual 3D world. The receding horizon real-time optimal control algorithms proposed in Chapter 6 will be employed in the framework. Through this application, we see an example of using autonomous switched system to model and solve a multi-agent network control problem. We also see the needs for the real-time algorithms established in this thesis and how they can be used in a non-abstract setting. More importantly, in this chapter we have established a novel optimization-based pilot decision support framework that is useful for the robotics and multi-agent network control community.

For this application, we aim to provide a framework, so that a multi-agent network can be controlled by a single human operator (a pilot in a remote location). The framework is designed to send useful information (computed by real-time optimal control algorithms) to the pilot, hence the name "decision support". However, the framework is specifically designed to only suggest the decision support and not force them. Furthermore, we design the framework to provide the pilot with critical information and total command over the network of agents, but without overloading the pilot with data. Hence, through this work, we also explore the problem of the balance of interaction between human and computers. A number of features useful for the proposed surveillance mission scenario are also developed, including threat detection and avoidance.

In this application, we produced software for the pilot to interact with the network and control the network remotely, as well providing optimization-based feedback to the pilot. All of these interactions take place in a user-friendly Graphic User Interface (GUI). A third party open-source (Player/Gazebo) simulation environment is used to generate a 3D environment where physical interactions of the UAVs can be simulated realistically. Furthermore, this simulation environment allows for a seamless transition from simulation to robotic hardware so that the system may be implemented on real UAVs in the future.

We first introduce the motivation for this application, and show how a multi-agent network can be modelled as an autonomous switched system. We then detail the software we developed to realize the proposed pilot decision support framework.

7.1 Introduction

One challenge facing coordination and deployment of unmanned aerial vehicles (UAVs) today is the amount of human involvement needed to carry out a successful mission. Currently, control and coordination of UAVs typically involves multiple operators to control a single agent. The aim of this chapter is to invert this relationship, enabling a single pilot to control and coordinate a group of UAVs. Furthermore, decision support is provided to the pilot to facilitate effective control of the UAV team. In the scenario envisioned in this chapter, the human operator (the pilot) is operating along-side a team of UAVs. The pilot communicates with the UAV team remotely and controls the UAV team to execute a surveillance mission.

An important aspect of this is the question of how much the pilot should interact with the UAV team and how much aid should be provided to the pilot without overloading the pilot with data and support. We address this issue by allowing two major modes of operations, namely *autonomous mode* and *pilot controlled mode*. In both of these modes, the UAV team is controlled in a leader-follower manner, and the leader UAV is assigned by the pilot, where the followers are positioning themselves with respect to the other UAVs in the network. In the autonomous mode, the leader UAV executes the mission without intervention of the pilot. At any time, the pilot is allowed to take over and directly control the leader vehicle. Hence, the pilot can interrupt the mission to investigate an area or avoid certain threats. The pilot can also release control of the UAV, and the UAV team automatically resumes the execution of the given mission.

The problem of controlling multiple agents in a coordinated fashion to achieve a set of goals such as maintaining desired formations, ensuring coverage of an area, or selection of the best leader for a group of agents, has received considerable attention during the last decade. In particular, multiple approaches have been developed to allow the inclusion of a human operator in the multi-agent system. In [1],[2],[3], and [4], the human operator provides commands to all the agents in the network by either total control of the robot, or supervisory control which modifies the behaviors of the robots. In either case, the same commands are provided to all the agents in the network simultaneously. Another work ([51]) developed a central interface to display the information of a swarm of mobile robots. In [26], the effect of information and decision aids provided to a human operator is examined in an abstract setting to address the question of how much interaction with the human should be provided.

The main novelty of the work in this chapter lies in the decision support aid provided to the pilot in the form of leader selection mechanisms. During operation of the system, the pilot can reconfigure the UAV network topology by assigning a different leader vehicle. When the system is in autonomous mode of operation, a desired leader selection is provided to the pilot. This selection is computed by a receding horizon, real-time optimal control algorithm that evaluates relative merits of different leader selections towards execution of the mission. It should be noted that we always insist on including the human operator in the loop for making high-level control decisions.

The support system developed here is designed with the philosophy in mind that the aid and support is provided but not forced. The mission implemented to demonstrate the pilot decision support system is a surveillance mission, and an a priori mission plan is generated at configuration time using pilot input. Once the mission is planned, it is executed autonomously. The pilot can take control at any time, during which the mission is paused. This is expected since the pilot may choose to investigate an area of interest or avoid certain threats. The overview of the pilot decision support platform is shown in Figure 25. The arrows represents information and interactions between different components of the framework.

Information and support is designed to be delivered to the pilot in a visually appealing way so that data can be understood and acted upon quickly. Hence, the central component of the system is a graphical user interface (GUI). An example of the GUI during operation of a mission is shown in Figure 26. All data and information are congregated and exchanged at the GUI, including local information about the individual UAV agents and decision support computed by the optimal control module. This interface also provides the mean of interaction between the pilot and UAV agents. The pilot selects and controls the leader vehicle by interacting with the GUI. In addition, the GUI displays data (location, altitude, etc.) of the network graphically so that information can be easily absorbed by the pilot. The background map shown on the GUI corresponds to the environment of the UAVs, which is a virtual 3D world constructed in a Player/Gazebo simulation environment, as shown in Figure 27. This simulation environment is described more in detail in the next section.

Even though this chapter focuses on a single-leader, multiple-followers scenario,

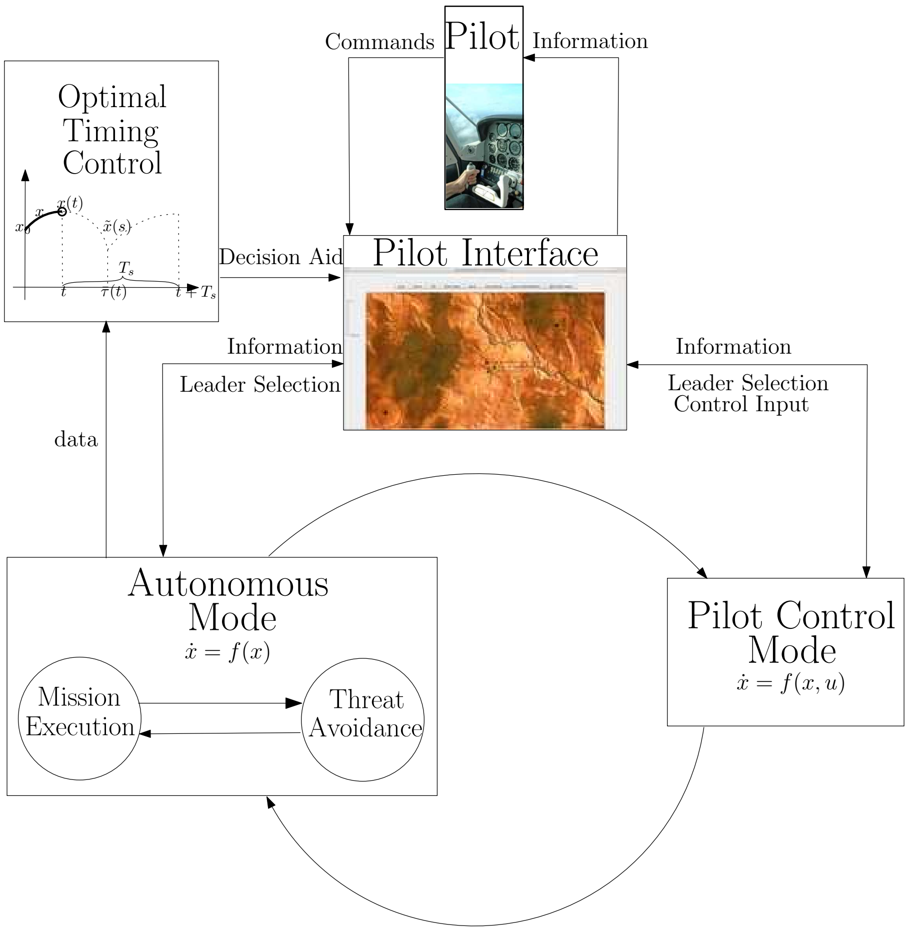


Figure 25: Pilot decision support platform architecture and different modes of operation

the platform introduced in this chapter can be generalized to a system with multiple leaders and groups. A multi-leader framework may create additional challenge for a single pilot to control at same time. The logistical challenge the pilot may face in terms of attention span for multiple groups is a different issue that is not addressed in this chapter. This is the reason why this chapter is focused on a single-leader framework. In terms of scalability, the computation effort involved for making optimal leader selections scales linearly with increasing number of agents. However, the communication required for the network may scale up to quadratically with the number of agents.

The platform supports autonomous mode and pilot control mode of operation.

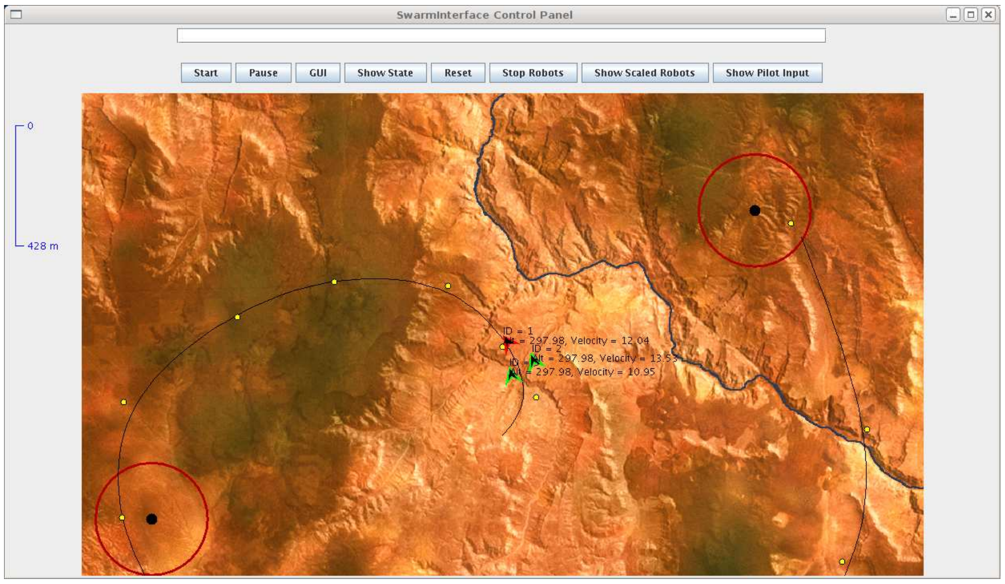

SwarmInterface Control Panel
Start
Pause
GUI
Show State
Reset
Stop Robots
Show Scaled Robots
Show Pilot Input
0
428 m
ID = 1
Alt = 297.98, Velocity = 12.04
ID = 2
Alt = 297.98, Velocity = 13.53
Alt = 297.98, Velocity = 10.95

Figure 26: GUI of the pilot decision support framework.

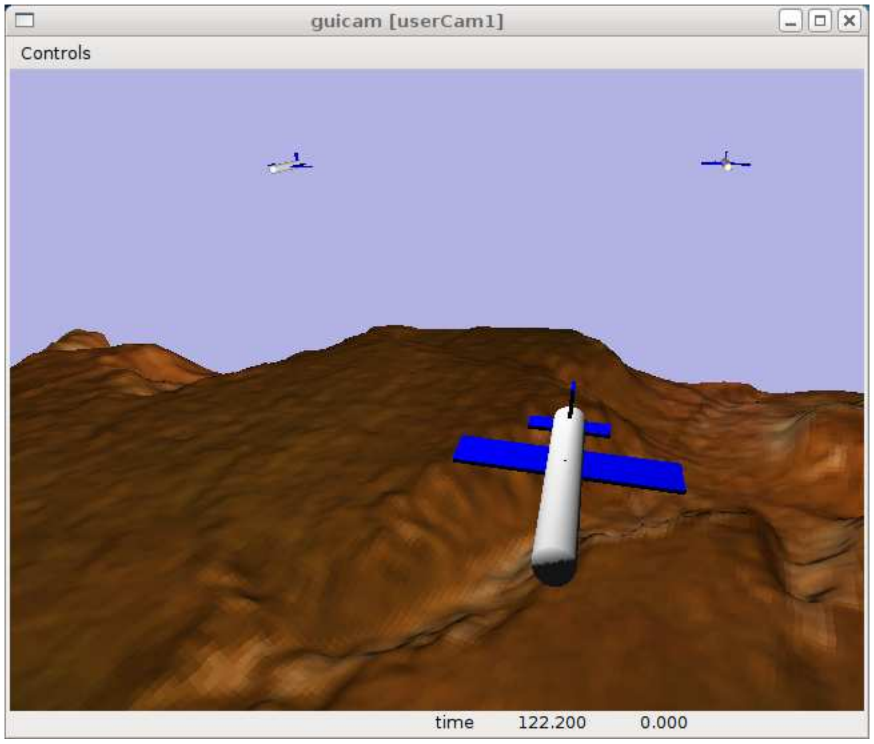


Figure 27: UAVs in Player/Gazebo 3D Simulation Environment.

The pilot can switch the system between these two modes at any time. For the majority of time in autonomous mode, the autonomous controller is designed to execute the surveillance mission. This mode of autonomous operation is referred to as the *mission execution mode*. There may be situations when an imminent threat is detected by local sensors on the UAVs, in which case the system switches to the *threat avoidance mode*, in which threat avoidance is carried out in higher priority. Furthermore, the optimal timing control module is designed to provide decision aid based on state information when the system is in mission execution mode. Every time the system switches to the mission execution mode, the decision aid is recomputed in a real time fashion. These modes of operations and the optimal control module are detailed in the following sections.

7.1.1 Autonomous Mode

One important goal of the proposed platform is to provide autonomous operation of the UAV team for a given mission. In autonomous operation, it is vital to choose a proper formation and topology for the multi-agent network in response to changes in the environment or human intervention. For example, reference [27] suggests using a spread out formation in open space, and a closely grouped formation in tight area. In the proposed framework, the UAV team is designed to be controlled in a leader-follower fashion. This has been an increasingly important approach to control multi-agent network systems and it is proven to be successful for many tasks (see [47] for example). Using the leader-follower control approach, the topologies of the network depends on the identity of the leader agent. After the pilot releases control, it may be necessary to re-evaluate the network topology so that a better candidate vehicle may be chosen as the leader. To this end, it is natural to characterize the network as a hybrid system that switches among a number of topologies (see [10] for a similar idea).

For the clarity of presentation, we consider a UAV team consisting of three individual UAVs. One of the UAVs is designated as the leader vehicle. The remaining UAVs act as followers, and each follower maintains a proper distance with respect to the leader and the other follower. Thus, in this UAV network, it is possible for the pilot to switch between 3 distinct subsystems, each defined by designating a different agent as the leader. At the planning level, the UAVs will be modelled as holonomic robots, and the state of the system can then be defined as $x = [x_1^T, x_2^T, x_3^T]^T$, where $x_i \in \mathbb{R}^3$ is the position of ith robot in the network. Since the mission will be specified as evolving on the plane, it is useful to defined the operator $Px_i = [x_{i1}, x_{i2}]^T$.

For the sake of implementation, the mission of the UAV network is defined as a surveillance task, and the goal of the mission is to provide coverage for an area. One way to achieve this task is to create a plan at configuration time using either a path planner or input from the pilot. The controller of the UAV network is designed to navigate through this path. This path is denoted as $p(t) \in \mathbb{R}^2$. We let the path to be generated by first laying down a set of way-points on the map of the area, then computing a smoothing spline curve based on the way-points. The smoothing spline guarantees the continuity of $p(t)$, while balancing the curvature of the path and the distance from the path to the given way-points.

Let the identity of the leader be l, where $l \in \{1, 2, 3\}$. We let the dynamics of the networked system in autonomous and mission execution mode be given by:

$$\begin{aligned} \dot{x}_l(t) &= \begin{bmatrix} \gamma(p(t) - Px_l(t)) \\ 0 \end{bmatrix} \\ \dot{x}_i(t) &= \alpha \sum_{j \neq i} (||x_i(t) - x_j(t)|| - k)(x_j(t) - x_i(t)), \\ & \quad i = \{1, 2, 3\} \backslash \{l\}. \end{aligned} \tag{163}$$

The leader controller is thus a simple proportional compensator with gain γ that drives the leader agent to the desired path. Note that the $\dot{x}_{i3}(t) = 0, i = 1, 2, 3$, because we assume that the UAVs fly in a constant altitude. We ensure the success of the surveillance task by designing the controller for the leader UAV so that it follows the path. This way, at least one UAV in the network is regularly covering the area over the given path. Note that the dynamics (163) represents an autonomous switched system, where there are three sub-systems and each sub-system employs a different leader.

The followers execute a weighted consensus dynamics with a predefined gain α. This controller is designed to maintain a fixed distance k from the follower to the other two UAVs, driving the UAV network to a triangular formation. The vertical component is moreover kept constant, to be used only when directly controlled by the pilot, or when avoiding threats. An example of the UAV network in formation with the leader tracking a path is shown in Figure 26. In the autonomous mode, the leader vehicle is colored red, and the followers are colored green.

In addition to mission execution, the proposed platform also provides threat detection capabilities. When an external threat is perceived by sensors, the UAVs must work to avoid the threat while navigating nominally along the defined path. This mode of operation within autonomous mode is referred to as the threat avoidance mode. Within the threat avoidance mode, the dynamics of the agents are determined by the behavior arbitrator of each agent. The behavior arbitrator generates the appropriate behavior so that each agent moves away from the threat while performing its individual role in a weighted fashion. Once the threat is perceived to be out of range, then the system returns to the mission execution mode. This switching behavior is illustrated in the autonomous mode module in Figure 25.

7.1.2 Optimal Timing Control

The objective of the optimal control module is to compute optimal switching times that minimize a cost function, which is formulated as a measure of the progress of the surveillance task. As a result, this decision aid is only provided when the pilot decision support system is in autonomous and mission execution mode. The switching times information is then fed-back to the pilot, and it is up to the discretion of the pilot on whether or not to switch according to the decision aid. For the pilot decision support framework, the network is reconfigured constantly during system run-time, and the switching times need to be constantly re-computed on-line and updated. Furthermore, since the final mission time can be long and optimal control requires simulation of the state trajectory, we use the receding horizon approach established in Chapter 6, in which a look-ahead window is selected, and the merits of switching to different leaders are evaluated over this horizon.

Consider the system in autonomous mode, where it can be represented by a switched system via (163). Define a switching sequence $\bar{q} = \{1, 2, 3\}$, where in mode i agent i is the leader. This mode sequence is assumed to be updated on-line by either switching (reducing the number of modes) and mode insertion (inserting a mode at the end of the time horizon). Hence, $q_0 = \{1, 2, 3\}$ serves as an initial condition to the algorithm. The mode insertion procedure is described in Chapter 2 and an online mode scheduling algorithm was discussed in Chapter 3. The control variable of the system is the switching time vector $\bar{\tau}(t)$ that is updated by the receding horizon algorithm (Algorithm 6.1 and 6.2). The task of the optimal control model is to minimize a time-dependent cost-to-go functional over a short time window of length T_s.

Since we define the surveillance task in terms of tracking a given path, the instantaneous measure of task progress can be formulated as the average distance of the UAV network to the path. This average distance is the distance between the centroid of the network to the path, and we use $centroid(x(t))$ to denote the centroid of the network. Hence, the real time optimal control problem is that of minimizing the centroid of the UAV team to the given path over the sliding window. This is a cost-to-go functional since it depends on the true time t, and is evaluated over a state trajectory over the time interval $[t, t + T_s]$. This state trajectory has to be simulated since it happens in the future. This future trajectory is denoted by $\tilde{x}(s)$, where $s \in [t, t + T_s]$ denotes the simulated time. The dynamics of the simulated trajectory is identical to the real system (defined in (163)), with the initial condition $\tilde{x}(t) = x(t)$.

As such, the cost-to-go functional is given by

$$J(t, x(t), \bar{\tau}(t)) = \int_t^{t+T_s} L(||centroid(\tilde{x}(s)) - p(s)||_2^2)ds.$$

This performance measure ensures that we select the UAV that minimizes the average distance between the UAV network to the given path. Let $q(t)$ denote the current mode at time t. As the algorithm updates the switching time on-line, when $\tau_{q(t)} = t$, we send a mode transition suggestion to the pilot (from $q(t)$ to the next mode in the sequence). Since the pilot may or may not abide to this suggestion, the real-time algorithm continues to run according to pilot's command. If the pilot make any mode transitions, then we update $\bar{q}$ and $\bar{\tau}(t)$ accordingly. If all modes are switched, then we insert a mode based on the insertion gradient computed by the insertion gradient formula (14) for the modes $q = 1, 2, 3$. The Algorithm 6.1 and 6.2 is then restarted.

7.1.3 Pilot Control Mode

As previously mentioned, at any time during the mission, the pilot may take control of the leader vehicle. In this mode of operation, the dynamics $\dot{x} = f_l(x)$ of the subsystem corresponding to l-th UAV being the leader of the UAV network becomes:

$$\dot{x}_l(t) \quad = \quad u(t),$$

where $u(t)$ is the control input (direction and speed) supplied by the pilot. The followers in this mode use the exact same dynamics as the mission execution mode (maintaining a distance k from each other), but the leader is controlled directly by the pilot.

The pilot enters the mode by clicking on the pilot virtual joystick as shown in the lower-right corner in Figure 28. The mouse click is used to generate the direction, and the speed of the vehicle can be adjusted by the virtual dial on the joystick panel. When the pilot is in control of a UAV, the vehicle is colored blue. If the pilot wishes to release control to autonomous mode, the pilot needs to click on the *Clear* button.

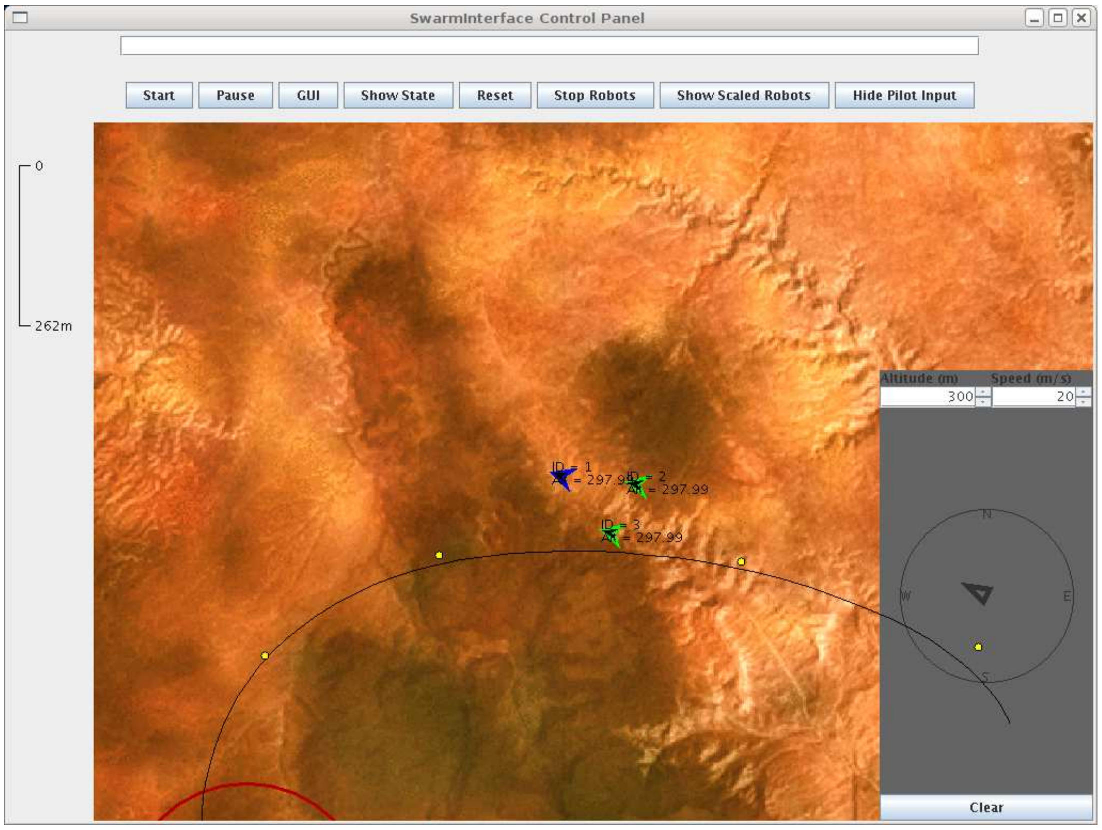


Figure 28: GUI when in pilot control mode of operation

7.1.4 System Logic

Since all the components of the system architecture are described, we are ready to explain the logic flow of the pilot decision support system for a surveillance mission. At every cycle (time-step), the system goes through a decision tree as shown in Figure 29, and switches to a different mode if necessary. The human pilot always have total control in decision making, including leader selection and movement of the UAV team. Furthermore, in autonomous mode, threat avoidance has higher priority than mission execution. Therefore the system switches to mission execution only when there is no threat perceived. It is moreover only during mission execution that the decision support will be computed and deliver to the pilot. This ensures that computational resources are not wasted. When the decision support is provided, it is up to the discretion of the pilot on whether or not to abide by the computed suggestion.

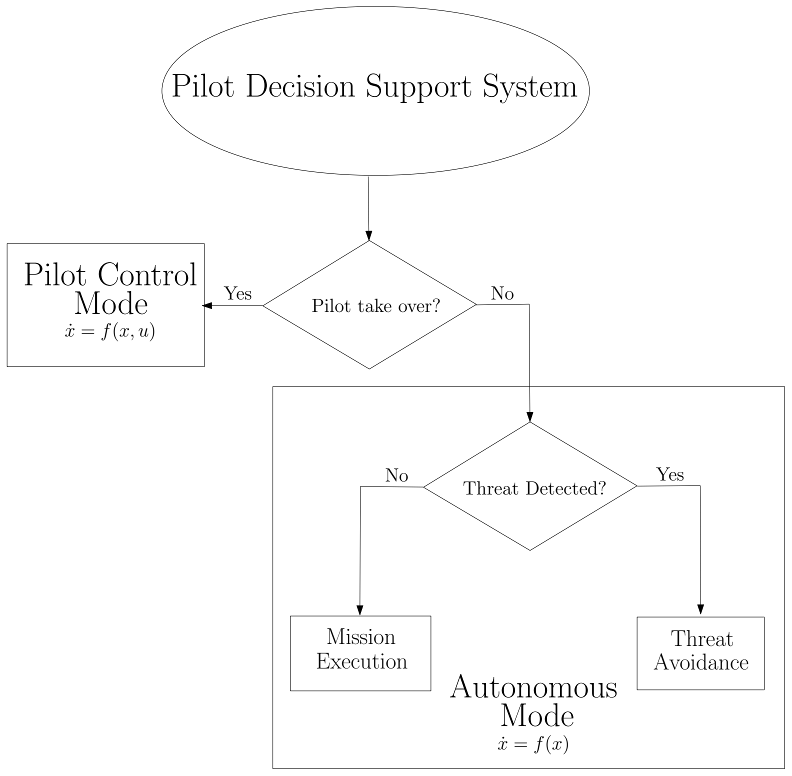


Figure 29: Logic of the pilot decision support system at each cycle.

7.2 Implementation

The pilot decision support system is implemented in a modular and extensible way. A third party open-source (Player/Gazebo) simulation environment is used to generate a 3D environment where physical interactions can be simulated realistically. Furthermore, this simulation environment allows for a seamless transition from simulation to robotic hardware so that the system may be implemented on real UAVs in the future.

7.2.1 Agent Implementation

The UAV agents are implemented using a modular, behavior-based design. Primitive and integrated perceptual information from a variety of sources are directed through a set of behaviors which calculate output preferences for control set-points in 3 dimensions: translational curvature within the x-y plane (κ), translational velocity within the x-y plane (v), and altitude (z). The output from the individual behaviors is combined in a behavioral arbitration layer, resulting in final κ, v and z set-points for the vehicle's low-level controllers. Figure 30 depicts the overall internal information flow within each agent.

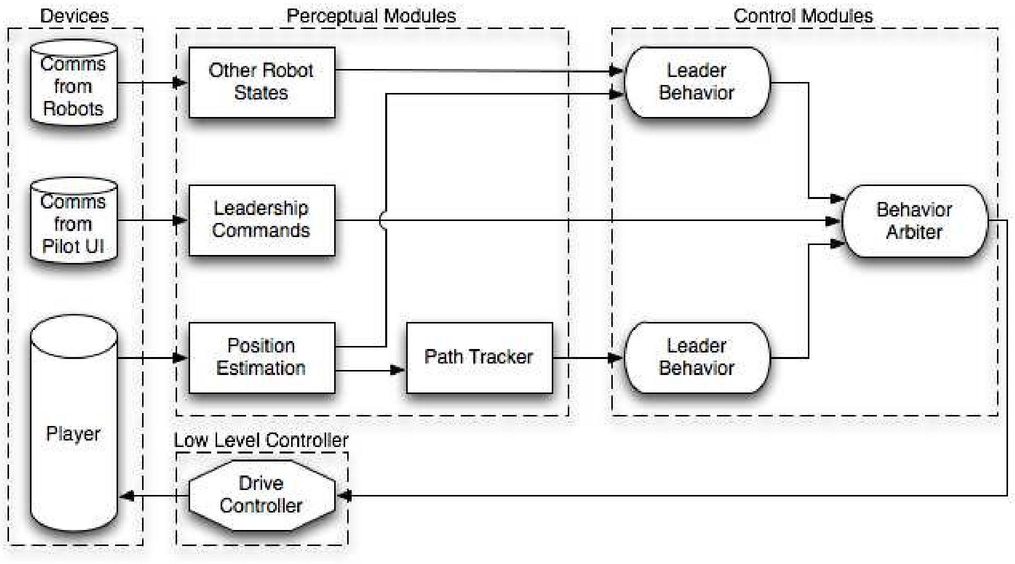


Figure 30: Data flow architecture for each UAV agent.

7.2.1.1 Perceptual Sources

Each robot in the simulation is implemented as an independent agent, acting either as a leader robot or a follower robot, according to input from the pilot. The robots act on information from several sources. These sources include:

- A priori plan - As described in Section 7.1.1, a plan is generated at configuration time for the UAVs to follow, which leads them through the simulated environment. The plan is represented by a time-parameterized smoothing spline curve based on a set of way-points provided by the pilot.
- External Perception - Via an interface with the simulation, the robots are able to observe their own relative and global positions. Additionally, they are aware

of features within the environment, including external threats, that must be avoided.

- Inter-robot communication - The robots are able to communicate their internal states, including whether they are currently acting as a leader or follower robot, with each other. The robots are also able to communicate their global positions to aid in the formation maintenance.

- Pilot input - The human pilot may communicate to the robots, choosing one to act as leader within the group, and, optionally, taking remote control of the leader.

7.2.1.2 Behavioral Control

The UAVs navigate using a behavior-based control system based on the DAMN architecture [60] and expanded upon in [65] and [69]. In this architecture, each behavior represents a particular interest of the agent (e.g. avoid threats, follow the plan, follow the pilot's input). The behaviors' output is expressed as preferences, or votes, across a set of discrete options for the next control set-point. Arbitration between these behaviors is performed by finding the option with the highest preference from a weighted sum of all behaviors.

In addition to expressing preferences across a set of possible κ values, each behavior may set a maximum allowable v and a minimum allowable z, given each κ value. Once a κ value is chosen by the arbiter, the minimum of the maximum allowable v's associated with the chosen κ is set as the translational velocity set-point. Likewise, the maximum of the minimum allowable z's associated with the chosen κ is set as the altitude set-point. The κ, v and z set-points are passed on to low-level controllers, which control the actuation of the vehicle to achieve the commanded set-points.

The set of available behaviors is made up of four distinct behaviors:

- Leader Behavior - At any given time, only one robot may act as a leader. The leader's task is to follow the a priori plan, using a simple proportional compensator, as described in Section 7.1.1.

- Follower Behavior - The follower robots act to maintain an appropriate distance from the leader and from each other by weighted consensus, as described in Section 7.1.1.

- Threat Avoidance Behavior - All robots work individually to avoid detected external threats. Threat detection is implemented using a local sensor model. The threat avoidance behavior only expresses a preference if an external threat is detected by the local sensor model.

- Pilot Guidance Behavior - The pilot may choose to take remote control of the leader robot. The pilot guidance behavior follows the input given by the pilot and described in Section 7.1.3.

7.2.2 Pilot Interface

The graphical user interface (shown in Figure 26) is implemented separately from the simulation environment, and it communicates with each agents via TCP/IP. It also communicates with the optimal control module to send state information and receive decision support. The GUI retains a copy of the mission (seen as the black curve in Figure 26), as well as initial way-points set by the pilot to generate the mission plan (yellow dots in Figure 26). The roles of the UAV agents are distinguished by color. In autonomous mode, the leader UAV is red. In pilot control mode, it is blue. Followers are always green, unless one is suggested as leader, in which case it becomes black. Known threat locations are displayed on the GUI as black dots in red circles.

7.2.3 System Implementation

Since each robot is implemented as an independent agent, system-wide organization and communication must be defined. Figure 31 depicts this system organization. At the top is the pilot interface. Through this module the pilot may monitor the state of the robots, select a leader robot, and, if he chooses, control the motion of the leader robot. The pilot input is communicated to each robot. Each robot takes this information in consideration, along with state information communicated from the other robots. Control commands are calculated and communicated via Player to the Gazebo simulation. State information is communicated from the simulation to the robots, which, in turn, communicate their state information to the pilot interface. Because each module is implemented as its own process, and communication implemented via TCP/IP, the system can be distributed over multiple computers, just as it would be in physical implementation.

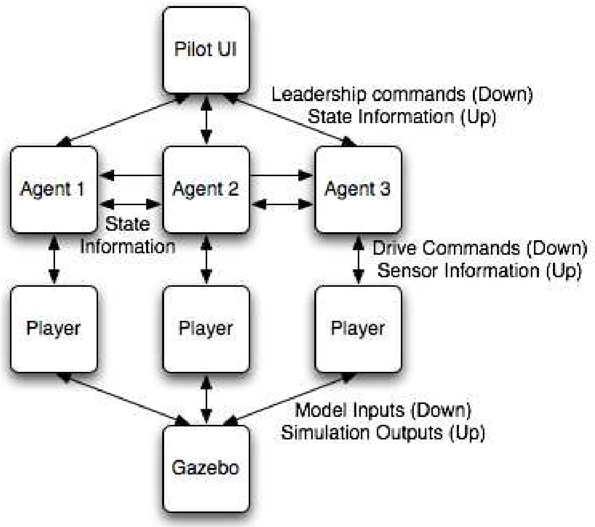
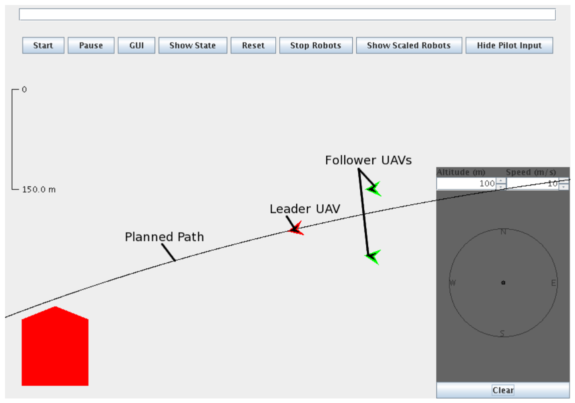


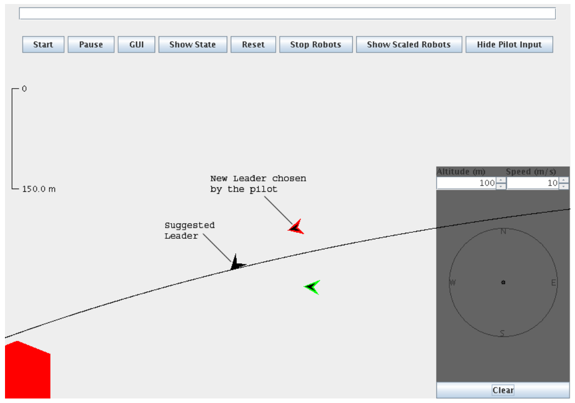
Figure 31: Data flow architecture for the experimental system.

7.3 Results

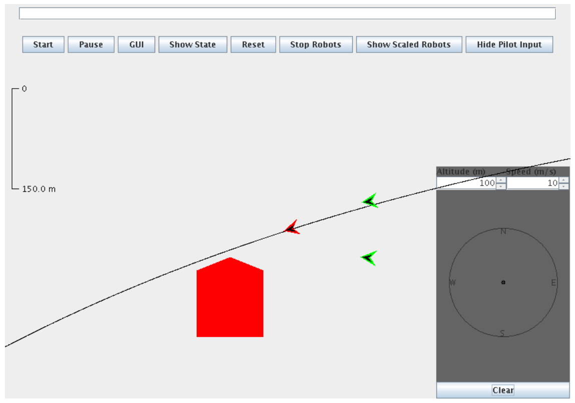
Demonstrations of the above described system are organized to highlight 4 features. Note that in all of the figures, background images are removed to show the results more clearly.



(a) The formation of UAVs follow the planned path.



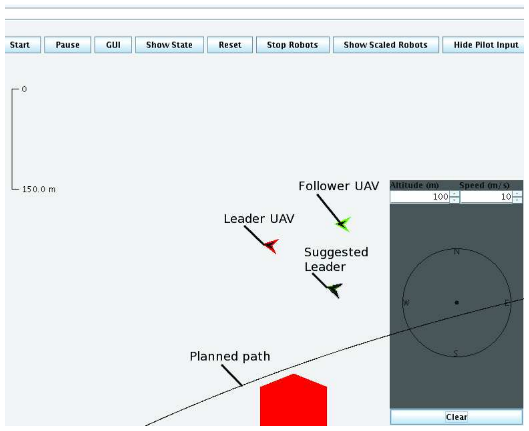
(b) The pilot chooses a new leader.



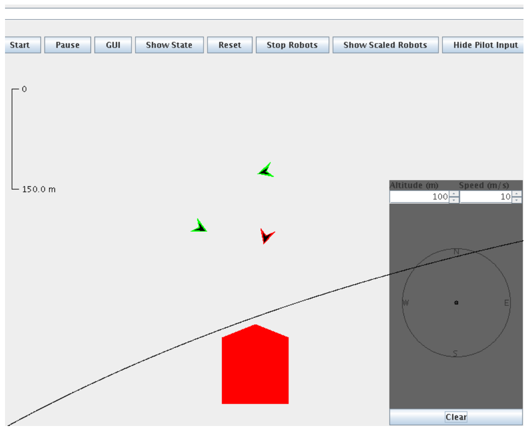
(c) The new leader takes its position at the front of the formation.

Figure 32: Demonstration of the pilot designating a new leader for the formation of UAVs.

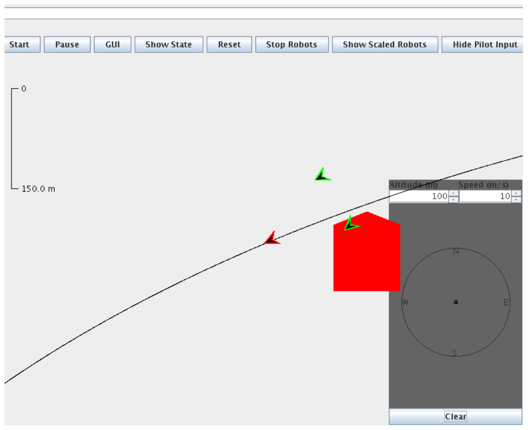
Figure 32 demonstrates the pilot's ability to designate a leader UAV from the team of UAVs. Figure 32(a), depicts the pilot interface showing a team of 3 UAVs navigating along a predefined route. The designated leader is in front (colored red), while the other two UAVs follow behind (colored green). In Figure 32(b), the pilot selects one of the two follower UAVs to become the new leader. Immediately, the selected UAV moves to take its place at the front of the formation, while the former leader moves to take its new place as a follower in the formation. Figure 32(c) shows the new steady state behavior with the newly designated leader in the front of the formation.



(a) The formation of UAVs is off the planned path. The pilot decision support system suggests a new leader to get back to the path faster.



(b) The pilot takes the suggestion of the decision support system, selecting the suggested UAV to be the new leader.

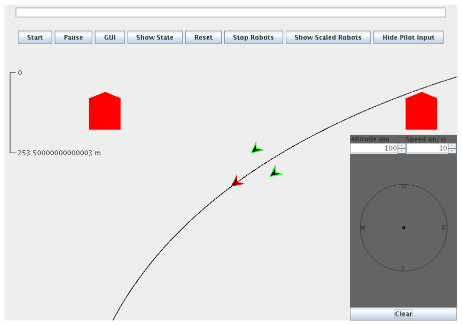


(c) The formation moves back to the planned path.

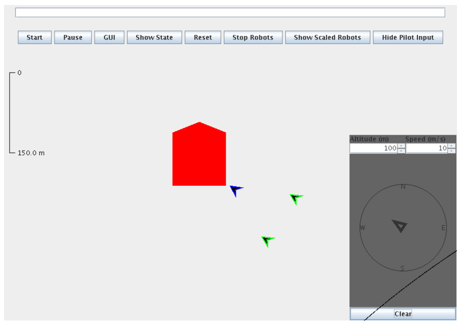
Figure 33: Demonstration of the decision support system aiding the pilot by suggesting a new leader

Figure 33 shows the pilot decision support system in action. In Figure 33(a), the

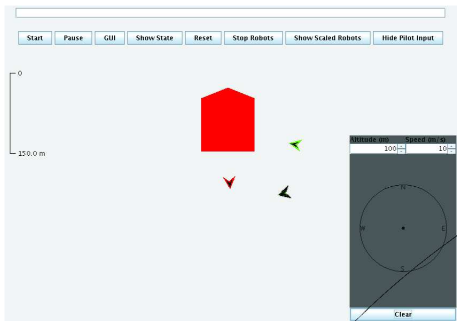
UAVs are in formation but offset from their commanded route. In this state, the UAV lowest in the screen would make the best leader, as evaluated by the pilot decision support system. This is communicated by coloring the center of the UAV in question black. Figure 33(b) shows state of the UAVs immediately after the pilot has selected the suggested UAV to be the new leader. The new leader moves to lead the other UAVs toward commanded route. Finally, Figure 33(c) shows the UAVs in steady state formation shortly thereafter.



(a) The UAVs move along the planned path. An area of interest to the pilot lies to the northwest of the formation.



(b) The pilot remote controls the leader to the area of interest.

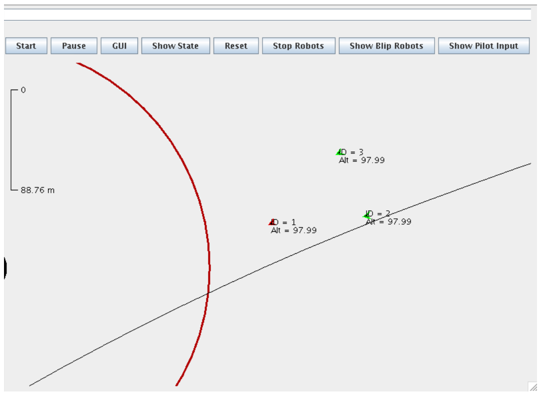


(c) The pilot returns the leader to autonomous control. The formation moves back to the planned path.

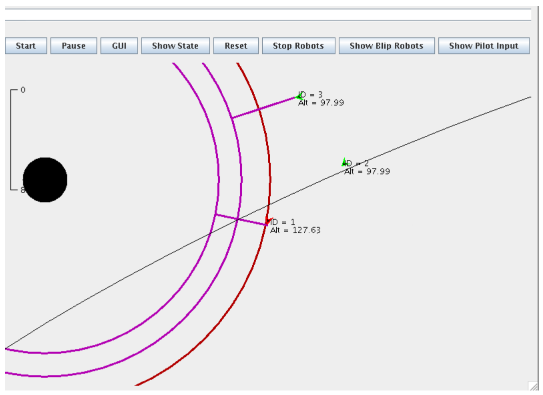
Figure 34: Demonstration of the pilot exploring an area of interest by remote controlling the leader UAV.

The affordance for the pilot to manually navigate the leader UAV is demonstrated by Figure 34. Figure 34(a) shows the formation of UAVs following their commanded route. An area of interest to the pilot (marked by the red polygon) is to the northwest of the formation. To explore the area of interest, the pilot uses the virtual joystick (shown in the bottom right corner of the interface) to pilot the leader UAV, as shown in Figure 34(b). The UAV under the pilot's control is marked blue. The other two UAVs continue to operate autonomously, albeit as followers. In this manner, the pilot is essentially in remote control of the entire formation. Once the pilot is satisfied with the exploration achieved, the leader UAV is returned to autonomous operation. Figure 34(c) shows the fully autonomous formation returning to it mission.

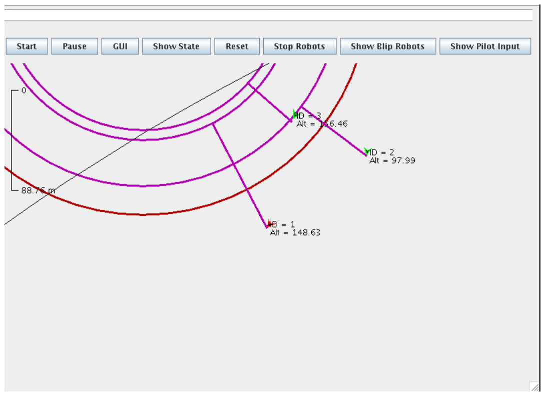
The threat avoidance behavior is highlighted in Figure 35. Figure 35(a) shows the formation of UAVs following their commanded route. An external threat, marked by the solid circle, is to the southwest of the formation, just off-screen. The threat range is modeled as a spherical region around the center. The range of the threat in the x-y plane at ground level is shown by the open circle around the threat. As the UAVs approach the threat, each UAV independently detects the threat. The detection is shown in Figure 35(b) by the open circles connected to each UAV. The radius of these circles represents the range of the threat in the x-y plane at the current altitude of each UAV. Figure 35(c) shows the UAVs navigating around the threat. The UAVs not only steer around the threat within the x-y plane, but also gain altitude to reduce the deviation needed within the x-y plane.



(a) The formation of UAVs navigating along the prescribed plan. An (as yet undetected) external threat is shown to the southwest of the formation, represented by a closed circle, just off-screen. The range of the external threat is modeled as a spherical region. The threat range within the x-y plane at ground level is shown as an open circle around the threat.



(b) The UAVs detect the threat. The open circles connected to each UAV represent the range of the threat within the x-y plane at the UAV's current altitude.



(c) The UAVs avoid the threat by both steering around it within the x-y plane, and gaining altitude to reduce the required deviation within the x-y plane.

Figure 35: Demonstration of the external threat avoidance capability of the UAV agents.

REFERENCES

[1] R.C. Arkin. Motor Schema-Based Mobile Robot Navigation. *International Journal of Robotics Research*, Vol.8 No. 4. August 1989, pp 92-112

[2] R.C. Arkin, Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, April 1992.

[3] R.C. Arkin. Reactive control as a substrate for telerobotic systems. *Aerospace and Electronic Systems Magazine*, IEEE Volume 6, Issue 6, June 1991 Page(s):24-31

[4] R.C. Arkin and K. Ali. Integration of reactive and telerobotic control in multi-agent robotic systems. *From animals to animats 3: Proc. Third International Conference on Simulation of Adaptive Behavior*, pages 473-478, 1994.

[5] R. Arkin. *Behavior-Based Robotics*. Cambridge, Massachusetts: MIT Press, 1998.

[6] L. Armijo, Minimization of Functions Having Lipschitz Continuous First-Partial Derivatives, *Pacific J. of Mathematics*, Vol. 16, pp. 1-3, 1966.

[7] S.A. Attia, M. Alamir, and C. Canudas de Wit, "Sub Optimal Control of Switched Nonlinear Systems Under Location and Switching Constraints", in *Proc. of the 16th IFAC World Congress*, Prague, Czech Republic, 2005.

[8] H. Axelsson. Optimal Control of Switched Autonomous Systems: Theory, Algorithms, and Robotic Applications. *PhD thesis*. Department of Electrical and Computer Engineering, Georgia Institute of Technology, United States of America, 2006.

[9] H. Axelsson, M. Egerstedt, and Y. Wardi. Optimal Switching Surfaces in Behavior-Based Robotics. *IEEE Conference on Decision and Control*, pp. 1954-1959, San Diego, CA, Dec. 2006.

[10] H. Axelsson, A. Muhammad, and M. Egerstedt. Autonomous Formation Switching for Multiple, Mobile Robots. *IFAC Conference on Analysis and Design of Hybrid Systems*, Sant-Malo, Brittany, France, June 2003.

[11] S. Azuma, M. Egerstedt, and Y. Wardi. Output-Based Optimal Timing Control of Switched Systems. *Hybrid Systems: Computation and Control*, Springer-Verlag, pp. 64-78, Santa Barbara, CA, March 2006.

[12] T. Balch, and R.C. Arkin. Behavior-based formation control for multiagent robot teams, *IEEE Transactions on Robotics and Automation*, December, 1998.

[13] R. Bellman. Dynamic Programming. *Dover*, 2003

[14] A. Bemporad, F. Borrelli, and M. Morari. Piecewise Linear Optimal Controllers for Hybrid Systems. *Proc. of the American Control Conf.*, pp. 1190-1194, 2000.

[15] A. Bemporad, F. Borrelli, and M. Morari. On the Optimal Control Law for Linear Discrete Time Hybrid Systems. *Hybrid Systems: Computation and Control*, pp. 105-119, 2002.

[16] D. Bernstein. Matrix Mathematics. *Princeton University Press*, 2007.

[17] B. Bethke, M. Valenti, and J. How. UAV Task Assignment. *IEEE Robotics and Automation Magazine*, Vol. 15, pp. 39-44, 2008.

[18] M. Boccadoro, Y. Wardi, M. Egerstedt, and E. Verriest. Optimal Control of Switching Surfaces in Hybrid Dynamical Systems. *Journal of Discrete Event Dynamic Systems*, Vol. 15, No. 4, pp. 433-448, Dec.2005.

[19] M.S. Branicky, Multiple Lyapunov functions and other analysis tools for switched and hybrid systems, *IEEE Transactions on Automatic Control*, vol.43, no.4, pp.475-482, Apr 1998

[20] M.S. Branicky, V.S. Borkar, and S.K. Mitter. A Unified Framework for Hybrid Control: Model and Optimal Control Theory. *IEEE Trans. on Automatic Control*, Vol. 43, pp. 31-45, 1998.

[21] R. Brockett. Hybrid models for motion control systems. *Perspectives in Control*, H. Trentelman and J. C. Willems, Eds, Birkh, Boston, pp. 2954, 1993.

[22] R. Brockett. Stabilization of Motor Networks. *IEEE Conf. on Decision and Control*, pp. 1484–1488, 1995.

[23] T.H.J. Collett, B.A. MacDonald, and B.P. Gerkey. "Player 2.0: Toward a Practical Robot Programming Framework". In *Proceedings of the Australasian Conference on Robotics and Automation* (ACRA 2005), Sydney, Australia, December 2005.

[24] D. Corona, J. Buisson, B. De Schutter, and A. Guia. Stabilization of switched affine systems: An application to the buck-boost converter. *American Control Conference*, pp. 6037-6042, 2007

[25] D. Corona, A. Giua, and C. Seatzu. Stabilization of switched systems via optimal control. In *Proceedings 16th IFAC World Congress*, Prague, The Czech Republic, 2005.

[26] M.L. Cummings. Human Supervisory Control of Swarming Networks, *2nd Annual Swarming: Autonomous Intelligent Networked Systems Conference*, June 2004.

[27] J.P. Desai, J. Ostrowski, and V. Kumar. Controlling Formations of Multiple Mobile Robots. Proceedings of the 1998 *IEEE International Conference on Robotics and Automation*, May 1998.

[28] C. R. Dohrmann and R. D. Robinett. Dynamic programming method for constrained discrete-time optimal control, *Journal of Optimization Theory and Applications*, Vol. 101, No. 2, pp. 259-283, 1999.

[29] X. Ding, Y. Wardi, D. Taylor, and M. Egerstedt. Optimization of Switched-Mode Systems with Switching Costs. *American Control Conference*, Seattle, WA, June 2008.

[30] X. Ding, M. Powers, and M. Egerstedt, and R. Young. An Optimal Timing Approach to Controlling Multiple UAVs. *American Control Conference*. St. Louis, MO, USA, June 2009.

[31] X. Ding, M. Powers, and M. Egerstedt, R. Young, and T. Balch. Executive Decision Support. *IEEE Robotics and Automation Magazine*, Vol. 16 No. 2, pp. 73-81, June 2009

[32] X. Ding, A. Schild, and M. Egerstedt, and J. Lunze. Real-time Optimal feedback Control of Switched Autonomous Systems. To be presented at *IFAC Conference on Analysis and Design of Hybrid Systems*, Zaragoza, Spain, Sept. 2009.

[33] X. Ding, Y. Wardi, and M. Egerstedt. Adaptive Optimal Timing Control of Hybrid Systems. *Mathematical Theory of Networks and Systems*, Blacksburg, VA, July 2008.

[34] X. Ding, Y. Wardi, and M. Egerstedt. On-line Adaptive Optimal Timing Control of Switched Systems. To be presented at *IEEE Conference on Decision and Control*, Shanghai, China, Dec. 2009.

[35] X. Ding, Y. Wardi, and M. Egerstedt. On-Line Optimization of Switched-Mode Dynamical Systems. *IEEE Transactions on Automatic Control.*, To appear.

[36] J. Dunn and D. Bertsekas. Efficient dynamic programming implementations of newton's method for unconstrained optimal control problems. *Journal of Optimization Theory and Applications*, Vol. 63, No. 1, pp. 23-38

[37] M. Egerstedt, S. Azuma, and Y. Wardi. Optimal Timing Control of Switched Linear Systems Based on Partial Information. *Nonlinear Analysis: Theory, Methods & Applications*, 2006.

[38] M. Egerstedt, and X. Hu. A hybrid control approach to action coordination for mobile robots. *Automatica*, Vol. 38, pp. 125130, 2002.

[39] M. Egerstedt, Y. Wardi, and H. Axelsson. Transition-Time Optimization for Switched-Mode Dynamical Systems. *IEEE Trans. on Automatic Control*, Vol. AC-51, pp. 110-115, 2006.

[40] D. Flieller, J.P. Louis, and J. Barrenscheen. General Sampled Data Modeling of Power Systems Supplied by Static Converter with Digital and Analog Controller. *Mathematics and Computer in Simulation*, Vol. 46, pp. 373-385, 1998.

[41] R. Goebel, R. Sanfelice, A. Teel. Hybrid dynamical systems. *IEEE Control Systems Magazine*, Vol. 29, No. 2, pp. 2893, 2009

[42] R. Ghaemi, J. Sun, and I. Kolmanovsky. Neighboring extremal solution for discrete-time optimal control problems with state inequality constraints, *in Proc. American Control Conference*, pp. 3823-3828, June 2008,

[43] A. Guia, C. Seatzu, and C. Van der Mee. Optimal Control of Switched Autonomous Linear Systems. In *Proc. 40th IEEE Conf. on Decision and Control*, pp. 2472-2477, 2001.

[44] S. Hedlund and A. Rantzer. Optimal Control of Hybrid Systems. *IEEE Conf. on Decision and Control*, pp. 3972-3977, 1999.

[45] D. Hristu-Varsakelis. Feedback Control Systems as Users of Shared Network: Communication Sequences that Guarantee Stability. *IEEE Conf. on Decision and Control*, pp. 3631–3631, 2001.

[46] B.Hu, X.Xu, A.N.Michel, P.J.Antsaklis. Stability analysis for a class of nonlinear switched systems. *Proc. 38th IEEE Conf. on Decision and Control*, pp.4374-9, Dec 1999.

[47] M. Ji, A. Muhammad, and M. Egerstedt. Leader-Based Multi-Agent Coordination: Controllability and Optimal Control. *American Control Conference*, pp. 1358-1363, Minneapolis, MN, June 2006.

[48] C.Y.Kaya, J.L.Noakes J.L. Computational Method for Time-Optimal Switching Control. *Journal of Optimization Theory and Applications*, Vol. 117, No. 1, pp. 69-92, 2003.

[49] B. Lincoln and A. Rantzer. Optimizing Linear Systems Switching. *IEEE Conf. on Decision and Control*, pp. 2063–2068, 2001.

[50] C. McCabe and USGS. *Grand Canyon Terrain.* [Online]. Available: http://www.cc.gatech.edu/projects/large_models/
gcanyon.html

[51] J. McLurkin, et. al. Speaking Swarmish: Human-Robot Interface Design for Large Swarms of Autonomous Mobile Robots, *AAAI Spring Symposium*, March 28, 2006

[52] N. Mohan, T.M. Undeland, and W.P. Robbins. *Power Electronics: Converters, Applications and Design*, 2nd Edition, John Wiley & Sons, New York, NY, 1995.

[53] J. Moon, Y. Wardi,and E. Kamen. Optimal release times in single-stage manufacturing systems with finite production inventory. *IEEE Conference on Decision and Control*, pp. 25062511, December 2002.

[54] P. E. Moraal and J. W. Grizzle, “Observer Design for Nonlinear Systems with Discrete-Time Measurements”, *IEEE Trans. on Automatic Control*, Vol. 40, pp. 395–404, 1995.

[55] R Olfati-Saber, J. Fax, and R. M Murray. Consensus and Cooperation in Networked Multi-Agent Systems. *Proceedings of the IEEE*, Vol. 95, pp. 215-233, 2007

[56] R. Olfati-Saber and R. M. Murray. Distributed Structural Stabilization and Tracking for Formations of Dynamic Multi-Agents. *Conference on Decision and Control*, 2002

[57] E. Polak, *Optimization Algorithms and Consistent Approximations,* Springer-Verlag, New York, New York, 1997.

[58] A. Rantzer and M. Johansson. Piecewise Linear Quadratic Optimal Control. *IEEE Trans. on Automatic Control*, Vol. 54, pp. 629-637, 2000.

[59] H. Rehbinder and M. Sanfirdson. Scheduling of a Limited Communication Channel for Optimal Control. *IEEE Conf. on Decision and Control*, pp. 1011–1016, 2000.

[60] J. Rosenblatt, DAMN: A Distributed Architecture for Mobile Navigation, *Journal of Experimental and Theoretical Artificial Intell.*, v. 9, n. 2, pp. 339-60, 1997.

[61] A. Schild, X. Ding, M. Egerstedt, and J. Lunze, Design of optimal switching surfaces for switched autonomous systems. *IEEE Conf. on Decision and Control*, Shanghai, China, 2009

[62] M.S. Shaikh and P. Caines. On the Optimal Control of Hybrid Systems: Optimization of Trajectories, Switching Times and Location Schedules. In *6th International Workshop on Hybrid Systems: Computation and Control*, 2003.

[63] M.S. Shaikh and P. Caines. On Trajectory Optimization for Hybrid Systems: Theory and Algorithms for Fixed Schedules. *IEEE Conf. on Decision and Control*, pp. 1997–1998, 2002.

[64] H.J. Sussmann. Set-Valued Differentials and the Hybrid Maximum Principle. *IEEE Conf. on Decision and Control*, pp. 558–563, 2000.

[65] J. Sun, T. Mehta, D. Wooden, M. Powers, J. Regh, T. Balch, and M. Egerstedt. Learning from Examples in Unstructured, Outdoor Environments, *Journal of Field Robotics*, Vol. 23, No. 11/12, pp. 1019-1036, Nov/Dec. 2006

[66] C. Tomlin, J. Lygeros, and S. Sastry, A game theoretic approach to controller design for hybrid systems. *IEEE Transaction on Automatic Control*, Vol. 88, pp. 949-970, July 2000.

[67] G. Walsh, H. Ye, and L. Bushnell. Stability Analysis of Networked Control Systems. *American Control Conf.*, pp. 2876–2880, 1999.

[68] Y. Wardi, X. Ding, M. Egerstedt, and S. Azuma. On-Line Optimization of Switched-Mode Systems: Algorithms and Convergence Properties. *IEEE Conference on Decision and Control*, New Orleans, LA, Dec. 2007.

[69] D. Wooden, M. Powers, M. Egerstedt, H. Christensen, and T. Balch. A Modular, Hybrid System Architecture for Autonomous, Urban Driving, *Journal of Aerospace Computing, Information, and Communication*, Vol. 4, No. 12, pp. 1047-1058, Dec. 2007.

[70] X. Xu and P. Antsaklis. Design of Stabilizing Control Laws for Second-order Switched Systems. *Prof. 14th IFAC World Congress*, Jul 1999.

[71] X. Xu and P. Antsaklis. Optimal Control of Switched Autonomous Systems. *IEEE Conf. on Decision and Control*, pp. 4401–4406, 2002.

[72] X. Xu and P.J. Antsaklis. Optimal Control of Switched Systems via Nonlinear Optimization Based on Direct Differentiations of Value Functions. *Int. J. of Control*, Vol. 75, pp. 1406-1426, 2002.

VITA

Xu Chu Ding is pursuing his Ph.D degree at the Georgia Institute of Technology, where he received his B.S degree in 2004 and M.S degrees in 2007 in electrical and computer engineering. He is currently working as a graduate research assistant at the Georgia Robotics and Intelligent Systems Laboratory. His research interests include real time optimal control of hybrid systems and multi-agent networked systems.